Tidal Layered Discovery Guide

Contents

Layering discovery techniques

This book will cover all the discovery steps you will need to perform for your cloud migration project.

Cloud migration is the process of moving your data, applications, and other elements to the cloud. However, the path to the cloud can be long and painful without proper planning and execution. By following Tidal's six discovery layering techniques, you will be migrating to the cloud with ease!

{% include image.html file="layered_discovery_expanded.png" caption="Tidal' layered approach to application discovery, for cloud migration." %}

Six Steps To Discovery Bliss

1) Import Your Spreadsheets

If your already have some data collected in spreadsheets, the first step to begin your cloud migration project is importing a spreadsheet of Virtualization Clusters, Servers, Databases Instances and Applications. Tidal's importer will guide you through mapping your columns to our fields, create your own fields and even make associations between dependencies if you have captured these.

NB: See additional ways on importing your applications and servers in the API docs.

---

2) Integrate Your Hypervisors

Once you have imported your data, you can begin to synchronize your inventories via `tidal sync servers`. Tidal sync supports many server inventory management tools such as VMWare and HyperV with more possible via scripting (ask us[1]).

If you have VM Ware's vSphere, `tidal sync vsphere` will handle everying with just read-only credentials required.

Scheduling your sync:

It is useful to setup a cron job or Windows Scheduled Task for this process, and we recommend synchronizing your inventories no more than once per day.

This will keep your resource inventory up to date and show you usage trends over time in the *Analyze* feature.

---

[1] https://tidalcloud.com/contact

3) Aggregate Server Usage Statistics

Tidal provides you with a simple and effective way to gather machine statistics[2] (RAM, Storage, CPU allocations and usage) from Windows and UNIX/Linux server environmens. In Windows, we use WinRM to Invoke-Command across your servers, and for *NIX we leverage ansible. Both of these approaches output JSON which is securely sent to your Tidal instance using the tidal command.

Checkout this guide for a quick and clean approach to gathering server usage statistics. See the machine_stats[3] repository for more implementation details.

NB: Feel free to fork the repo and modify to suit your needs, or to show your security team and give them comfort. This extensibility and transparency is core to our approach.

---

4) Fingerprint Web Applications

a.) The initial step in your cloud journey is discovering what you have. It can be hard to remain informed about all the domains and applications hosted in your environment, which is why we created the `tidal discover` command. With your customized *Discovery Plan* you can obtain both private and public domains within your datacentres in under 60 seconds.

This tidal-tools guide contains examples for creating your own *Discovery Plan* to scan multiple networks and DNS services.

b.) With a list of domains in hand, the next step is to *analyze* the applications hosted on these domains.

`tidal analyze web` will fingerprint the technology on both your internet sites and intranet applications behind your firewall in seconds, *without needing to install agents.* Whether you have 1 or 1 million end points, Tidal Tools centralizes the data gathered into our platform for you to analyze further and plan with, simplifying your application centric cloud migration.

For detailed information and steps on analyzing your domains, be sure to checkout this guide.

---

5) Analyze Your Databases

Analyze all of your databases in minutes and *measure* the migration difficulty.

After running `tidal analyze db your_config.yml` against your databases (see guide), you will understand which database features in your Oracle,SQL Server, MySQL, or PostgreSQL installations make it difficult to adopt cloud-native database offerings and also identify which applications are connecting to your databases.

With over 100 unique characteristics considered, comparisons are made with the data platforms available in the cloud(s) you are migrating to which provide you with *data-driven insights* for planning your cloud migration.

Follow the steps in the guide, and you will be able to quantify the difficulty in migrating your database from Oracle to PostreSQL; or from SQL Server to AWS Aurora etc.

---

[2]https://github.com/tidalmigrations/machine_stats
[3]https://github.com/tidalmigrations/machine_stats

6) Analyze Your Source Code

Finally, to find the applications which will migrate more easily to cloud-native technologies you can analyze your source code and rank your applications by *Cloud Readiness*.

Doing this for each of your custom applications which have a *Transition Type* of Refactor or Replatform will give you the data needed to prioritize your application migrations. To analyze your source code, you need the `Application ID`, to be logged in with tidal-tools and a copy of the source code checked out.

You can find the `Application ID` in the URL bar when looking at an application. e.g. if I'm loking at an application in Tidal, the URL will show `https://[your workspace].tidal.cloud/apps/111/overview` Here, `111` is my `Application ID`.

I can now analyze the source code with:

```
cd /path/to/source-code
tidal analyze code --app-id 111 .
```

To find additional information about this feature, visit the guide on analyzing your source code.

The remaining will now go through the discvory process step by step. Starting with installing Tidal Tools.

Getting started with Tidal Tools

{% include note.html content="This page describes general installation and configuration instructions for Tidal Tools. If you are looking for specific AWS usage instructions please refer to Getting Started with Tidal Tools on AWS guide." %}

Here we outline how to get started working with Tidal Tools via the command line.

Downloading & Installing

Tidal Tools is a cross platform CLI utility that you can use to discover and analyze your applications. Easily install Tidal Tools[4] on your operating system.

If you are looking for a full walkthrough of the steps, you can watch a brief video on how to install it on Windows, Linux[5] or macOS[6].

If you'd prefer to use a cloud shell, you can watch how to get setup in the AWS[7], Azure[8] or Google[9] cloud shells.

Dependencies

If you plan to use the `tidal analyze code` command to analyze your source code or `tidal analyze db` to analyze your databases, you will need to install Docker Community Edition. To install Docker on your system you can check Docker's documentation directly[10]. Once installed you can verify it is installed and working correctly with the command `tidal doctor`.

Docker for Windows supports both Linux containers and Windows containers, however **Tidal Tools works when your Docker installation is set to use Linux containers**. Set up Docker for Windows to use Linux containers.

Why Docker?

You need to install Docker in order to complete the database analysis. This is because the analysis uses several system dependent software libraries, so by using Docker the analysis can use those libraries without you requiring to install the correct dependencies with the correct versions.

---

[4]https://get.tidal.sh
[5]https://www.youtube.com/watch?v=t86W7BokqGM
[6]https://www.youtube.com/watch?v=DyHT0mtPcfE
[7]https://www.youtube.com/watch?v=YpX64_uU-Oo
[8]https://www.youtube.com/watch?v=N_o_GSSY0zg
[9]https://www.youtube.com/watch?v=lHO5sk54ceo
[10]https://docs.docker.com/install/

Using Tidal Tools

To get started, from a new terminal or Powershell prompt, simply run:

```
tidal
```

To see what you can do with the tidal checkout some of our other articles about creating sync jobs or analyzing your applications via their URLs.

Connecting to the API

Once you have Tidal Tools installed you need to configure access to the API. Register for a free account[11] with Tidal to get the connection details.

There are several ways to authenticate with the Tidal API, we recommend the first one, `tidal login`, because your password is never persisted to disk.

Tidal Login command

To authenticate with the API type `tidal login` and follow the prompts. This should provide and store an access token for you that gives you access for 8 hours. Once it is expired, the user must login again and obtain another token.

We recommend that you utilise this command because it doesn't store your password. If you must store the password, you can authenticate using the methods below.

Alternative authentication methods

Configuration file

Alternatively, you can use the `tidal config` command to manually set your credentials.

Only your Tidal password and vSphere password are encrypted using AES 256 bit encryption.

For example, you can set your username, password and URL with the three following commands:

```
tidal config set tidal.email [set your email]
tidal config set tidal.password [set your password]
tidal config set tidal.url https://demo.tidal.cloud
```

Your credentials will be stored in a configuration file such as:

```
tidal:
  email: my_user_name_here
  password: my_secure_password_here
  url: https://my_instance_name_here.tidal.cloud
```

{% include note.html content="Your Tidal password and vSphere password are encrypted, using AES 256 bit encryption when stored in the configuration file. The other parts of the configuration file are not encrypted." %}

On macOS the config file is located: `$HOME/Library/Preferences/tidal/config.yaml`

On Linux systems it is: `$HOME/.config/tidal/config.yaml`

And on Windows it is: `%AppData%\tidal\config.yaml`

---

[11] https://get.tidal.cloud/

Environment Variables

You can specify your credentials as environment variables. If so, these variables need to be set:

- `TIDAL_EMAIL`
- `TIDAL_PASSWORD`
- `TIDAL_URL`

Dynamically

Alternatively you can pass in your credentials on each request as command line arguments using the following flags:

```
--tidal-email youremail@address.here
--tidal-password your_secure_password
--tidal-url https://yoursubdomain.tidal.cloud
```

Testing connectivity and authentication

Once you have set your credentials you can test your connectivity and authentication to the API with the ping command:

```
tidal ping
```

Using a Proxy

If you need to use a proxy, you can specify it with the `https_proxy` variable.

On macOS and Linux you would run:

```
export https_proxy=1.1.1.1:123
```

and on Windows:

```
set https_proxy=1.1.1.1:123
```

Of course replacing `1.1.1.1` with your proxy IP address and `123` with the port number.

Once you have this set, you should be able to run `tidal ping` successfully. Remember that you will need to set this for every new terminal session you start.

Import data from Excel

Tidal's importer will guide you through mapping your columns to our fields, creating your own fields, and even making associations between dependencies if you have captured these. When using the Tidal importer, it will create or update records depending on the names or host names uniqueness.

You can import your Excel spreadsheets into Tidal by visiting:

`https://your_workspace.tidal.cloud/import/`

Preparing your Spreadsheet

If you'd like to try the process itself but don't have a file yet, you can use this sample spreadsheet.

You should import your data in the following order:

1. Servers
2. Database Instances
3. Applications

This order is especially important if you are planning to import dependencies for your Databases and Applications. Ie. If your Application has a dependency on a server, you need to have that server imported first for the dependency to be imported correctly.

{% include note.html content="You can always update a dependency after importing too." %}

You should have one sheet or file per type of record.

There are several default columns values included for each record type, however if you have additional information on your inventory that would like to include, you can create custom fields for a record (Check step 3). Remember to add those fields before importing the data.

How to import your data

1. Select the data type you wish to import.

   {% include image.html file="import_step0.png" %}

2. Select the sheet in your file that you want to import data from.

   {% include image.html file="import_step1.png" %}

3. Map the columns of your sheet to the fields from the drop down. If you have a column that is not present as a field and you want to import it, you can add that field.

   {% include image.html file="import_step2.png" %}

4. Click submit to import the data.

{% include note.html content="If you are importing applications, the *name* field is required for each application in order to successfully import. For any of the other record types you are importing, there are no fields that are required in order for you to import them." %}

{% include tip.html content="The Tags option can be used to generate multiple tags and associate them with the record. You can use this track any type of categorical data. Additionally you can search and filter for records based on tags. Multiple tags can be associated based on one column of data, they are parsed and split on any commas (,)" %}

Importing Servers

There are no required fields when importing Servers.

Default fields: - Host Name *() - FQDN - Assigned ID - Zone - Role - RAM Allocated (GB) - RAM Used (GB) - CPU Count - CPU Name - Storage Allocated (GB) - Storage Used (GB) - Description - IP Addresses -> Address ()* - Database Instances -> Name *() - Environment -> Name ()* - Cluster -> Host Name (*) - Tags - Operating System Version - Operating System

(*) Associative fields. * *Host Name*, This field is used to identify the server and to create association with Databases and Applications, * *Cluster -> Host Name*, This field is used to indicate that the Server that belongs to a virtual machine cluster with *Host Name*. * *Database Instances -> Name*, If a server is hosting a Database (or multiple), this field should contain the name**(s) of such database(s). * *Environment -> Name*, This field is used to represent the server's environment.

Importing Databases

There are no required fields when importing Databases.

Default fields: - Name *() - Description - Database Size (MB) - Database Path - Server -> Host Name ()* - Environment -> Name (*) - Tags

(*) Associative fields. * *Name*, When you are importing other records, Applications and Servers, and you provide a set of dependent databases with them the Name field for Database Instances is used to identify those databases and associate the databases as dependencies for those other records. This is helpful when you have some existing dependencies, between Databases and Servers or Applications, in your Excel files and want to capture and leverage them. * *Server -> Host Name*, represents the server hosting a Database. This field should contain the Host Name** of such server. * *Environment -> Name*, This field is used to associate a Database to an environment.

Importing Applications

When importing Applications, only the **Name** field is required.

Default fields: - Name - Description - URLs - Source Code Location - Total Users - Revenue - Annual Hosting Costs - Annual Staff Costs - Uptime Requirements - Data Sensitivity - Frequency Of Deployments - PII - Legal Holds - COTS - Source Code Controlled - Continuous Delivery - Business Continuity Plan - Can Run On Linux - End Of Support Date - Servers -> Host Name *() - Database Instances -> Name ()* - Environment -> Name (**) - Customers - Tags - Technical Lead - Business Owner

(*) Associative fields. However, in order to properly map your inventory you must use the following fields. * *Servers -> Host Name*, represents the server hosting the Application. This field should contain the Host Name **of such server(s). * *Database Instances -> Name*, This field should contain the** Name** of the database(s) associated to the Application. * *Environment -> Name*, This field is used to associate an Application to an environment.

Sync with your hypervisors

Overview

As part of your Cloud Migration journey, it is very important to have a reliable inventory of your infrastructure. Tidal Tools is able to connect to a Hypervisor to synchronize your inventory of virtual machines from vSphere.

This guide will walk you through the steps and necessary configuration to properly integrate with vSphere and capture the following data points on its VMs.

- FQDN
- IP Addresses
- HostName
- AssignedID (InstanceUuid in vSphere)
- RAM Used (GB)
- RAMAllocated (GB)
- CPU Name
- CPU Count
- Storage Used (GB)
- Storage Allocated (GB)
- Operating System
- Operating System Version

Requirements

- Tidal Tools. You can check out the Getting Started with Tidal Tools guide on how to install it.
- vSphere Credentials.

Supported Versions

Tidal Tools is able to integrate with the following versions:

vSphere Version	Supported	Notes
5.5	Yes	When you set the vsphere.server you will likely need to include the default port, 443, in the url. Ex. 192.168.1:443
6.0	Yes	

vSphere Version	Supported	Notes
7.0	Yes	

Authentication

First, you need to login into your workspace. On your Terminal run the login command.

```
tidal login
```

Once you are logged on to the Tidal API, You can connect to your vSphere account. The following command will guide you through a set of prompts:

```
tidal login vsphere
```

1. vSphere Server - Either an IP address or a FQDN that resolves to the correct IP for the ESXi or vCenter instance
2. vSphere Username - A username that has read access to the vCenter or ESXi instance.
3. vSphere Password - Corresponding password for the user.
4. vSphere API Endpoint - The default is `/sdk`, It can be left as is, unless you know your setup does not use the default.
5. Use SSL/TLS? - The default is to use SSL (y), which can be left unless you know you can't use it.
6. Skip HTTPS certificate checks? - The default is to not skip (n), which can be left unless you know they need to be skipped.

After answering the above prompts, you will see the message:

```
Login to vSphere successful. Saving config file...Done!
```

Sync

Synchronizing your vSphere inventory with Tidal is simple with:

```
tidal sync vsphere

Loading data..
Transforming data..
Validating JSON...
Valid.
Logging in to server at https://demo.tidal.cloud as info@tidalcloud.com
Logged in.
Updating via API..
Update successful
```

You can easily split this into two commands to allow you to inject other metadata about your inventory, if needed. To do this simply use the get command to get your inventory. Modify it as you need and then you can use the sync command to upload your data.

1. `tidal get vsphere`, will fetch your vSphere inventory and output JSON (check it out!).

2. Modify the data as you like from Standard Input and then send it to Standard Output.

3. `tidal sync servers`, will take that JSON from STDIN and send it to Tidal.

```
tidal get vsphere | ./modify_script.rb | tidal sync servers
```

Repeat

You can easily set this to run periodically. The integration updates records if they already exist, or creates new records if they don't. Look at setting this command up as a cron job[12] that runs once per day.

Additional Configuration Options

The integration can be configured with several other methods besides `tidal login`. You can use these alternate approaches if they better suite your use case.

The other configuration options available are to set values in the configuration file (either interactively or manually) or set environment variables.

The credentials and configuration settings take precedence in the following order:

1. Environment Variables
2. Configuration file
3. Prompt in the terminal - If neither are present, you will be prompted to enter the required details and they are stored in the default configuration file.

1) Configuration File

Set Interactively

You can set your vSphere credentials with the following commands:

```
tidal config set vsphere.username [your username]
```

```
tidal config set vsphere.password [your password]
```

```
tidal config set vsphere.server 192.168.1.12
```

The set values are stored in the default configuration file.

{% include note.html content="Your configuration file is not encrypted. Only your Tidal password and vSphere password are encrypted using AES 256 bit encryption." %}

Set Manually

You can create a file, for example `config.yml`, with content similar to this:

```
vsphere:
  username: my_user_name_here
  server: 1.1.1.1
tidal:
  email: my_user_name_here
  url: https://my_instance_name_here.tidal.cloud
```

{% include note.html content="You can not manually set or edit any passwords in the configuration file. They are stored here only as encrypted values. To persist a password, it must be added by using the `tidal config set` command."%}

That can be used with a command: `tidal sync vsphere --config config.yml`

---

[12]https://www.digitalocean.com/community/tutorials/how-to-use-cron-to-automate-tasks-on-a-vps

tidal

16

2) Environment Variables

Another alternative is to set the following environment variables with the needed values and Tidal Tools will use them:

- VSPHERE_USERNAME
- VSPHERE_PASSWORD
- VSPHERE_SERVER
- VSPHERE_TLS
- VSPHERE_INSECURE

Troubleshooting

Login failing with error `expected element type <Envelope> but have <html>`

The reason behind this behavior is that provided vSphere login data references vCenter REST API[13] service which is exposed on `/rest` (versions `<=6.5`) or `/api` (versions `>6.5`) endpoint.

Instead, Tidal Tools CLI uses vSphere Web Services API[14] which is exposed on `/sdk` endpoint. This service is enabled by default on vSphere products for versions `>=v5.5`. If you have manually disabled this service, Tidal vSphere functionalities cannot be used.

How to login?

```
vsphere.server: FQDN (or IP)
vsphere.username: username@domain (or domain\username)
vsphere.api_path: /sdk (default path to vSphere Web Services API)
vsphere.tls: true (or false if using http)
vsphere.insecure: true (or false to skip certificate checks)
```

Additional context

1. vSphere Automation API[15] - provides REST API for managing and automating infrastructure operations of various vSphere services (includes *vCenter REST API* among other APIs).
2. vSphere Web Services API[16] - provides SOAP API for centralized management of vSphere infrastructure components (virtual machines, datacenters,...).

The difference between the APIs is that Automation API provides access to underlying service management, while Web Service API provides access to the management of individual components.

What is Syncing?

Syncing is a process that creates and updates your inventories in your Tidal Workspace. This is useful if you would like to update a lot of data across many different records. It can also be useful to keep your inventory up to date with an external date source, by continually updating your inventory on a recurring schedule.

---

[13] https://developer.vmware.com/apis/vsphere-automation/latest/vcenter/
[14] https://developer.vmware.com/apis/1355/vsphere
[15] https://developer.vmware.com/apis/vsphere-automation/latest/
[16] https://developer.vmware.com/apis/1192/vsphere

How do I sync my servers?

{% include note.html content="You will first need to have Tidal Tools installed and logged in[17] to your Workspace." %}

To upload your server inventory into your Tidal workspace you need to transform your data into JSON format.

This data can be passed as standard input to the `tidal sync servers` command and your servers data will be synchronized via the API.

The JSON document must have this specified format. The top-level key must be `"servers"`, with a value of an array. The array can consist of the various keys as shown below, describing the server to be synced. You can also include any other arbitrary fields in the key `"custom_fields"`.

{% include note.html content="If you pass data to Tidal Tools via standard input (ie. `cat x.json | tidal sync servers`) any custom fields presented will be automatically created if they don't already exist. If you provide the data to the command as an argument (ie. `tidal sync servers x.json`) you will be prompted to create any custom fields if they don't already exist." %}

```
{
  "servers": [
    {
        "host_name": "ewrfceapcfg03",
        "description": "This is a general description for this server.",
        "custom_fields": {
          "tcp_port": 11441,
          "database_software": "SQL 2008 SP3",
          "database_version": "10.0.5538.0"
         },
        "fqdn": "ewrfceapcfg03.com",
        "environment_id": 2,
        "assigned_id": "198",
        "zone": "Data",
        "ram_allocated_gb": 8,
        "storage_allocated_gb": 83.8,
        "storage_used_gb": 52.06,
        "cluster_id": 48337,
        "role": "Administrator",
        "cpu_count": 4,
        "ram_used_gb": 2,
        "virtual": true,
        "environment": "Production",
        "cluster": {
          "host_name": "rrfedfds"
        }
    }
  ]
}
```

The synchronization of your servers to Tidal can be performed with the following command:

```
tidal sync servers some_file.json
```

Any servers provided that already exist, based on their hostname, will be updated with the data provided in the JSON document. If no server with the given hostname exists, that server will be created with all the included data.

You can easily set this to run periodically so that your servers are synced on a daily basis and the data is up to date. This is a great resource[18] on setting the command as a cron job.

---

[17] https://guides.tidal.cloud/tidal-tools.html

[18] https://www.digitalocean.com/community/tutorials/how-to-use-cron-to-automate-tasks-on-a-vps

We recommend that you setup your inventories to sync *every 24 hours*, this will keep your resource inventory up to date and accurate over time.

Transforming your data

If your server inventory is not available in the required JSON format, don't worry we have you covered.

Suppose you have a **csv** file.

Below is a **sample** Ruby script *transform.rb* that will read the data within the file passed in as standard input, transform it to the JSON format above and output it as standard output.

```ruby
#!/usr/bin/env ruby
require 'json'
require 'csv'

def to_bool(str)
  if str == "true" || str == "yes"
    str = true
  else
    str = false
  end
  str
end

def transform(input)
    # convert input to proper csv format
    csv = CSV.parse(input,{ encoding: "UTF-8", headers: true, header_converters: :symbol, converters
    # convert csv to hashmap with key value pairs
    json = csv.map { |row| row.to_hash }
    data = {servers: []}
    json.each do |vm|
        props = {}
        props[:cluster] = {:host_name => vm[:cluster_host_name]}
        props[:custom_fields] = {:tcp_port          => vm[:custom_fields_tcp_port],
                                 :database_software => vm[:custom_fields_database_software],
                                 :database_version  => vm[:custom_fields_version]}

        props[:assigned_id]  = vm[:assigned_id].to_s
        props[:virtual] = to_bool(vm[:virtual])
        c = vm.merge(props)
        data[:servers].push c
    end
    # display data in pretty json format
    puts JSON.pretty_generate(data)
end
data = STDIN.read
transform data
```

The script above is an example of how to easily transform your data into the necessary JSON object. It can be altered to work with your data or it could be rewritten in any language of your choice.

Change the file permissions to make the script executable using:

```
chmod +x ./transform.rb
```

You can now use your script with a given CSV file and it will be synced to your account via the API. Run the following command:

```
./transform.rb < some_file.csv | tidal sync servers
```

How do I sync my servers in more automated fashion?

Instead of manually providing all of your servers details, you can utilize Machine Stats[19] utility to gather all the stats from your servers environment. Follow the Machine Stats installation and configuration steps available on the tool's GitHub page[20].

How do I sync other resources?

You also have the option of syncing your **Applications** and **Database Instances**.

Sync your Applications

You can sync your Applications with the following command:

```
tidal sync apps some_file.json
```

When importing your applications to the API, Tidal's sync tool will check for existing applications, based on their name, and update the changed data for those applications. If the given application to sync does not exist already, it will add that application to the Tidal API.

To synchronize your Application data, It must have the following JSON format.

```
{
    "apps": [
        {
            "name": "App_name",
            "custom_fields": {
                "technologies": "Approval Management System DB"
            },
            "description": "This is a general purpose application",
            "servers": [
                {
                    "host_name": "trpewrcapbiz02"
                }
            ],
            "database_instances": [
                {
                    "name": "my_app_db"
                }
            ],
            "urls": [
                {
                    "url": "https://approvalmanagementsystem.com"
                }
            ],
            "source_code_location": [
                "/filepath/location",
                "folder1/file1"
            ]
        }
    ]
}
```

---

[19]https://github.com/tidalmigrations/machine_stats
[20]https://github.com/tidalmigrations/machine_stats

{% include note.html content="This[21] is a similar script as the one above to transform your data into the necessary JSON object for applications." %} {% include note.html content = "There is also sample data[22] for applications." %}

Sync your Database Instances

You can sync your Database Instances with the following command:

```
tidal sync dbs some_file.json
```

When importing your database instances to the API, Tidal's sync tool will check for existing database instances, based on their name, and update the changed data for those database instances. If the given database instance to sync does not exist already, it will add it to the Tidal API.

To synchronize your Database Instances data, It must have the following JSON format.

```
{
  "database_instances" : [
    {
      "name": "720 TASK DB",
      "database_size_mb": 1870,
      "database_path": "C:\\system\\databases\\720_TASK_DB",
      "description": "This is a general description for this database instance. This database primar
      "environment": "production",
      "custom_fields": {
        "technologies": "Approval Management System DB"
      }
    }
  ]
}
```

---

[21] https://github.com/tidalmigrations/gists/blob/master/transformations/scripts/csv_transform.rb
[22] https://github.com/tidalmigrations/gists/blob/master/transformations/data/apps.csv

Web applications discovery, analysis and importing

Discover Your Applications

The first step in your cloud migration project is discovering what you have. Thousands of domains are registered on a daily basis and it can be hard to remain informed. Utilize the `tidal discover` tool with your customized Discovery Plan to obtain both private and public domains registered for your given datacentres.

- Scan multiple networks and DNS services with a *discovery plan*

  ```
  tidal discover my_plan.yml > my_urls.txt
  ```

{% include note.html content="In order to utilize this command, install DNS Tools[23]." %}

With this command, Tidal Discover will output a set of FQDNs for your defined discovery plan and store it in the file *my_urls.txt*.

Your Discovery plan is a YAML[24] file which can include three different ways that you want to scan your networks and DNS services. You may choose to provide a DNS service to extract information, a *named.conf* file for binary configuration, or a collection of zone files to be scanned and generate all the affected domains.

via DNS Service

An example of a discovery plan to obtain FQDNs by specifying a DNS Service.

The file *my_plan.yml* must be of the following format:

```
discovery:
  - name: Q9 Datacenter front-ends
    networks: 10.83.2.0/24
    tcp_ports:
      - 80
      - 443
    dns_service: aws
```

via Bind Configuration

An example of a discovery plan to obtain FQDNs by specifying a named.conf file.

The file *my_plan.yml* must be of the following format:

---

[23] https://dnstools.ninja/download/
[24] http://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html

```
discovery:
  - name: NYC Datacenter front-ends
    networks:
      - 10.83.3.0/24
      - 10.130.241.0/24
    tcp_ports:
      - 80
      - 443
    path_to_bind: "/etc/bind/named.conf"
```

via Zone files

An example of a discovery plan to obtain FQDNs by specifying a zone file.

The file *my_plan.yml* must be of the following format:

```
discovery:
  - name: Tokyo flat-network
    networks: 192.168.0.0/16
    tcp_ports:
      - 80
      - 443
      - 8080
      - 8443
    zonefiles: "~/tokyo_zones/*/**"
```

You may also choose to include all three of the ways in your Discovery plan like so:

```
discovery:
  - name: Tokyo flat-network
    networks: 192.168.0.0/16
    tcp_ports:
      - 80
      - 443
      - 8080
      - 8443
    zonefiles: "~/tokyo_zones/*/**"

  - name: NYC Datacenter front-ends
    networks:
      - 10.83.3.0/24
      - 10.130.241.0/24
    tcp_ports:
      - 80
      - 443
    path_to_bind: "/etc/bind/named.conf"

  - name: NYC Datacenter front-ends
    networks:
      - 10.83.3.0/24
      - 10.130.241.0/24
    tcp_ports:
      - 80
      - 443
    dns_service: aws
```

You can also combine all as the following:

```
discovery:
  - name: NYC Datacenter front-ends
    networks:
      - 10.83.3.0/24
      - 10.130.241.0/24
    tcp_ports:
      - 80
      - 443
    path_to_bind: "/etc/bind/named.conf"
    zonefiles: "~/path/to/my/zonefiles"
    dns_service: aws
```

Creating your Discovery Plan

Here is some brief information regarding the keys defined in the *my_plan.yaml* file:

Key	Information	Format
`networks`	One or more subnets that you want to include in the process.	Cidr block notation[25]
`name`	A friendly name for your network, e.g. "Tokyo DC-1 Front-End"	Text
`tcp_ports`	One or more TCP Ports that you frequently run web servers on and would like to interrogate: e.g. 80,443,8080,8443 etc.	Integer
`path_to_bind`	The location of a named.conf file[26] for a bind server configuration.	File Path
`dns_service`	Name of a DNS service to be analyzed with DNS tools, currently only "aws" service is avaliable which extracts information from Amazon Route 53 zones.	"aws"
`zonefiles`	The location of a zone file[27] which contains a list of DNS records with mappings between domain names and IP addresses.	File Path

{% include note.html content="`networks`, `name` and `tcp_ports` are required keys that you must include. Specify one or more of `path_to_bind`, `zonefiles` **or** `dns_service` in your Discovery file." %}

Be sure to verify the outputted FQDNs that you'd want to analyze.

Next Step

Having discovered your applications, here is a guide on analyzing your FQDNs. Tidal Analyze will review the outputted FQDNs and give you a detailed analysis on what technologies are being in use for each domain.

Tidal Analyze Web

Now that you have discovered your FQDNs for your specified Discovery Plan, the next step is to rapidly capture what technologies are in use, on which networks and with what DNS configuration.

---

[25] https://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing
[26] https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/5/html/deployment_guide/s1-bind-namedconf
[27] https://help.dyn.com/how-to-format-a-zone-file/

Tidal Analyze Web will fingerprint the technology on both your internet sites and intranet applications behind your firewall in seconds, without needing to install agents. Whether you have one or one million end points, Tidal Tools will centralize the data gathered in our platform for you to analyze.

Simplify your application centric discovery with Tidal Analyze Web.

Gathering your domains and URLs

Simply save a list of URLs or FQDNs in a text file and use that as input.

Or use the discovery command to scan your network and DNS to gather a list of relevant domains.

Analyzing FQDNs and URLs

Now that we have our relevant domains and URLs in *my_urls.txt*, we can analyze them with:

```
tidal analyze web my_urls.txt --upload
```

{% include tip.html content="You can also utilize the standard output of URLs and FQDNs from Tidal Discover as input. `tidal discover my_plan.yml | tidal analyze web --upload`" %}

{% include callout.html content="Are you running this behind a firewall or in a private network? No problem, drop the `--upload` flag, continue on and then checkout the Uploading to your account section below." type="primary" %}

Uploading to Tidal API

After recieving the results of Tidal Analyze Web, you may or may not be connected to the internet.

If you are so, you can import it to your Tidal account with the following flag:

```
tidal analyze web my_urls.txt --upload
```

If you aren't connected to the internet, you also have the option to save the results in a JSON file to import at a later time. This can be done with the following steps:

1. Run this command to run the analysis and save the results to the file *analyze_output.json*:

   ```
   tidal analyze web my_urls.txt --type json > analyze_output.json
   ```

2. Copy the file, *analyze_output.json* and install Tidal Tools, on a computer with internet access.

3. Login to Tidal Tools with `tidal login`

4. Run this command to upload your previously generated data to your Tidal Migrations account:

   ```
   tidal analyze web --upload-file analyze_output.json
   ```

Accessing your domains in the Tidal API

Once you have imported the results to the Tidal API, your domains will appear on the right hand side navigation bar under

Assess > **URLs**.

Troubleshooting

It looks like the server did not respond for 10 seconds

If you got this error message you can try to increase the waiting time using the `--timeout` flag and the amount of seconds (default is `10`). For example, `--timeout 30` sets the time waiting for the server to respond to 30 seconds.

Host discovery with Nmap

As part of your cloud migration journey, it is important to have all the tools at your disposal. To facilitate the discovery process, you can use Nmap[28] in addition to the other discovery methods[29].

With Nmap's powerful capabilities and good documentation[30], you can identify all the hosts you have running on your network, open ports, and running services. It is possible to discover those places which are often hard to reach.

Nmap is capable of producing its output in an XML file. It allows you to inspect the raw scan output before sending it to the Tidal API with Tidal Tools[31]. Once uploaded to the Tidal Platform, you will be able to visualize your network devices, track your complete server inventory, and build on this data with other discovery methods. This is how you can make informed decisions on your cloud migration path.

Using Nmap with Tidal Tools

By leveraging the power of **Tidal Tools**, you can send the output generated by Nmap to your Tidal account.

1. Install Tidal Tools[32]

2. Connect Tidal Tools and your Tidal account with tidal login[33].

3. Run Nmap with the flags of your choosing and save the output to an XML file. For example,

   ```
   nmap -sV -p80,443,8080,8443,1433,1521,27017 <ip-address/range> -oX my-network.xml
   ```

   *Note: the **-sV** flag will attempt to determine the version of the service running on port and the **-oX** specifies the output as an XML file. Want more scanning options?*

   {% include tip.html content="Be sure to replace with the CIDR range for the network you would like to scan, ex. `10.0.0.0/24`" %}

4. Run this Tidal Tools command to upload your previously generated Nmap output to your Tidal account `tidal sync nmap my-network.xml`

5. Head over to your Tidal account! (https://[your workspace].tidal.cloud/host-discovery)

Installing Nmap

Nmap ("Network Mapper") is a free and open source utility for network discovery and security auditing. Many systems and network administrators also find it useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.

---

[28] https://nmap.org/
[29] https://guides.tidal.cloud
[30] https://nmap.org/book/host-discovery-find-ips.html
[31] https://get.tidal.sh
[32] https://guides.tidal.cloud/tidal-tools.html
[33] https://guides.tidal.cloud/tidal-tools.html#login

You can find all the documentation[34] and instructions on how to download[35] Nmap to your environment on the official site[36].

Nmap usage

{% include callout.html content="Currently, Tidal supports the collection of Hosts (IP addresses), PTR records, their open ports, the ports status, the port protocol (TCP/UDP), and the services running in the port, including the version." type="primary"%}

Nmap offers a wide range of utilities and commands, such as Port scanning, Host discovery, Service and version detection to name a few. Here are some basic examples for how to do some nmap scanning.

{% include note.html content="Some commands require super-user privilege" %}

Target Specification

```
nmap 192.168.1.1-254            # Scan a range
```

Scan Techniques

```
nmap 192.168.1.1 -sS            # TCP SYN port scan (Default)

nmap 192.168.1.1 -sT            # TCP connect port scan (Default without root privilege)

nmap 192.168.1.1 -sU            # UDP port scan
```

Host Discovery

```
nmap 192.168.1.1-3 -sL          # No Scan. List targets only

nmap 192.168.1.1/24 -sn         # Disable port scanning. Host discovery only.

nmap 192.168.1.1-5 -Pn          # Disable host discovery. Port scan only.
```

---

[34] https://nmap.org/docs.html
[35] https://nmap.org/book/install.html
[36] https://nmap.org/

Database analysis

Not sure how ready you are to move to the cloud? With Tidal, you have the option to analyze the databases associated with your applications.

The analysis will calculate the difficulty of migrating your databases to each target platform, and give details on database features that may complicate the migration.

It is capable of analyzing **Oracle**, **SQL Server**, **MySQL**, and **PostgreSQL** databases. Providing analysis on migrating a database to a **variety of services on AWS, Azure, and Google Cloud**.

{% include note.html content="The entire analysis never queries or reads user or application data and does not collect database source code." %}

Supported Database Versions

Tidal Tools is able to analyze databases with the following versions:

Oracle	SQL Server	MySQL	PostgreSQL
Oracle Database 8i (8.1)	SQL Server 2005	5.5	8.3
Oracle Database 9i Release 2 (9.2)	SQL Server 2008	5.6	9.1, 9.2, 9.3, 9.4, 9.5, 9.6
Oracle Database 10g Release 2 (10.2)	SQL Server 2008R2	5.7	10
Oracle Database 11g Release 1 (11.1)	SQL Server 2012	8.0	11
Oracle Database 11g Release 2 (11.2)	SQL Server 2014		12
Oracle Database 12c Release 1 (12.1)	SQL Server 2016		13
Oracle Database 12c Release 2 (12.2)	SQL Server 2017		
Oracle Database 18c (18.0, 18.1)	SQL Server 2019		
Oracle Database 19c (19.0, 19.1, 19.2, 19.3)			

If you have a use case for a different version, definitely let us know at info@tidalcloud.com, we are always adding new capabilities.

Migration Complexity

The databases are analyzed to look for patterns and feature usage that may be difficult to migrate due to lack of support or compatibility in their new environment. The databases are analyzed based on their metadata, looking at specific schema objects that are used within your databases as well as the usage of proprietary features that will not be available in the target platforms.

For example, in Oracle databases, the Data Dictionary and AWR repository tables are read and analyzed. The scoring is calculated based on the type of attributes, features or schema objects that are used and the frequency of use throughout the database.

- Over 100 unique characteristics are considered
- Feature-fit is executed against all supported cloud data platforms.
- Migration difficulty score is calculated based on a weighted model

Getting Started

1. Before you can analyze a database, You must enable the Database Analysis feature for your account. To do so, go to your workspace Settings/Preferences (https://[your workspace].tidal.cloud/preferences). You will find the Database Analysis section at the bottom of the page.
2. Now that you have activated the Database Analyze feature, You need to install, configure and authenticate via Tidal Tools. Make sure you follow these guides.

   • How to install Tidal Tools.
   • Install Tidal Tools dependencies.
   • Make sure you can connect with your workspace using Tidal Tools.
   • As a last step, You should run the tidal doctor command to verify that your environment has been configured properly.

3. Tidal Tools will need a YAML configuration file with values and credentials needed to connect to your database. You can check this section for more information on how to create such a file.
4. Configure your Database to allow access to Tidal Tools. This section will walk you through the creation of a `tidal` user and granting the necessary permissions.
5. Perform the Database Analysis. There are 2 options to do so. You can find more details in this section

That is all, You should be able to see the results in your workspace within seconds.

Create your Database configuration file

For the most part, all supported Database engines share very similar configuration with Oracle being the exception.

Your YAML configuration file for Postgres, MySQL, and SQL Server must contain the following information.

   • `id` - The id of the database from your Tidal account. You can find it in the URL bar when looking at a database instance. ex. If you are viewing a database instance in Tidal, the URL will show https://[your workspace].tidal.cloud/database_instances/111/overview in this case 111 is the database instance ID.
   • `name` - A common name for your database could be the same or different from db_name, but this value is arbitrary and only for your reference.
   • `engine` - The database vendor, either `SQL Server`, `MySQL`, or `PostgreSQL`, it is not case sensitive.
   • `host` - The hostname of the server that the database is located on and is accessible via a network connection from your current device and location.
   • `port` - The port that the host has open and the database can accept connections on, for SQL Server it is `1433` or the port of the named instance, for MySQL it is `3306`, and for PostgreSQL the default port is `5432`.
   • `db_name` - The name of the database you are analyzing // or in the case of Oracle, the "service name".
   • `user` - A username to authenticate with the database, check the section below for more details about creating a user and granting permissions.
   • `password` - A password for the corresponding user.

When analyzing Oracle databases, the attributes in your configuration file must contain the following details.

   • `id` - The id of the database from your Tidal account. You can find it in the URL bar when looking at a database instance. ex. If you are viewing a database instance in Tidal, the URL will show https://[your workspace].tidal.cloud/database_instances/111/overview in this case 111 is the database instance ID.
   • `name` - A common name for your database.

30

- `analyze_workload` - When set to **true** workload analysis is enabled. Be sure to configure AWR data retention and verify licensing. For more information check the advance configuration section
- `engine` - **Oracle**
- `host` - The hostname of the server that the database is located on and is accessible via a network connection from your current device and location.
- `port` - The port that the host has open and the database can accept connections on, the default for Oracle is `1521`.
- `db_id_type` - Specifies the identifier type. Must be one of the following values:
    - **SID** if entering an Oracle System ID (SID) in the next field.
    - **Service Name** if entering a service name in the next field.
- `sid_or_service_name` - If you selected SID as your **db_id_type**, you should add the Oracle database SID, otherwise enter your Oracle service name.
- `user` - A username to authenticate with the database, check the section below for more details about creating a user and granting permissions.
- `password` - A password for the corresponding user.
- `is_pdb` - An Oracle Multitenant Pluggable Database (PDB). If set to true, You will need to provide credentials for the Container Database in following fields.
- `cdb_sid` - The Service ID of the container database.
- `cdb_user` - The common username of the container database user.
- `cdb_password` - The password of the container common database user.

The simplest way to create your configuration file is to use the `tidal analyze db init` command in Tidal Tools. The command will generate for you a databases.yaml file with a template with each one of the supported engines. All you will need to do is populate the file according to your needs.

{% include note.html content="It is best to use quotations, either double or single, around the values in the configuration file. To avoid special characters, : { } [ ] , & * # ? | - < > = ! % @ \ \n from being interpreted. Single quotes are safest, if the value has a single quote within it, you can include it by using a two single quotations, ie. `'my''string'` - will become `my'string`." %}

{% include tip.html content="Are you analyzing an Oracle Stanard Edition (SE) database or using a CDB database? Check out the advanced configuration below." %}

Configuring a User

To make the process as easy and secure as possible, you can create a user in your database dedicated to `tidal`. This will provide you full control and transparency on the permission Tidal Tools needs when performing the Database Analysis. Below you will find instructions and scripts for each supported Database. ### Oracle User This SQL script will create a `tidal` user and set the right permissions for you. Once downloaded, You should edit it and provide a secure password on the first line.

If you are using a CDB database you will also need to create a second user to access the CDB. You can do that with this script and also provide a secure password at the top:

```
CREATE USER c##tidal_comm_user IDENTIFIED BY "replace_this_with_secure_password" account unlock;
GRANT CREATE SESSION to c##tidal_comm_user;
GRANT select_catalog_role to c##tidal_comm_user;
```

SQL Server User

You can use this SQL script to create a `tidal` user and set the correct permissions. You should edit it and provide a secure password on the first line.

MySQL Server User

For MySQL you can create a user (`tidal`) with all the necessary permissions using the following statements:

```sql
CREATE USER 'tidal'@'%' IDENTIFIED BY 'replace_this_with_secure_password';
GRANT PROCESS,REFERENCES, SHOW DATABASES, SHOW VIEW ON *.* TO 'tidal'@'%';
GRANT SELECT ON sys.* TO 'tidal'@'%';
GRANT SELECT ON performance_schema.* TO 'tidal'@'%';
GRANT SELECT ON mysql.slave_master_info TO tidal;
GRANT SELECT ON mysql.slave_relay_log_info TO tidal;
GRANT SELECT ON mysql.user TO tidal;
```

In addition, use the following command that applies to your MySQL version. #### MySQL 5.x

```sql
GRANT SELECT ON mysql.proc TO tidal;
```

MySQL 8.x

```sql
GRANT SHOW_ROUTINE ON *.* TO 'tidal'@'%';
GRANT SELECT ON mysql.user TO 'tidal'@'%';
```

PostgreSQL Server User

Use the following commands to create a user (`tidal`) on PostgreSQL server.

```sql
CREATE USER tidal WITH PASSWORD 'replace_this_with_secure_password';

DO $$ DECLARE comm_rec RECORD;
BEGIN
    FOR comm_rec IN
            SELECT 'GRANT REFERENCES ON ALL TABLES IN SCHEMA '||schema_name||' TO tidal' AS comm
        LOOP
        EXECUTE comm_rec.comm;
    END LOOP;
END;
$$ LANGUAGE plpgsql;

GRANT REFERENCES ON ALL TABLES IN SCHEMA public to tidal;
```

For PostgreSQL versions **higher than 9** the following GRANTs should be also applied:

```sql
GRANT SELECT ON pg_catalog.pg_config TO tidal;
GRANT EXECUTE ON function pg_catalog.pg_config TO tidal;
GRANT SELECT ON pg_catalog.pg_proc TO tidal;
GRANT SELECT ON pg_catalog.pg_namespace TO tidal;
```

For PostgreSQL versions **10 and above** the following GRANTs should be also applied:

```sql
GRANT SELECT ON pg_catalog.pg_hba_file_rules TO tidal;
GRANT SELECT ON pg_catalog.pg_roles TO tidal;
GRANT pg_read_all_settings TO tidal;
GRANT pg_read_all_stats TO tidal;
```

After creating the user you will need to add the appropriate entry to the `pg_hba.conf`[37]. For example:

```
# TYPE   DATABASE    USER      ADDRESS       METHOD
host     all         tidal     0.0.0.0/0     md5
```

To apply the configuration changes to the running PostgreSQL server you will need to run `pg_ctl restart`[38].

---

[37]https://www.postgresql.org/docs/current/auth-pg-hba-conf.html
[38]https://www.postgresql.org/docs/current/app-pg-ctl.html

Perform the analysis

When analyzing a database (or multiple), You have two options.

1. Perform the database analysis and upload the result immediately to your workspace. This is Tidal Tools' default behaviour. To do so, all you need to do is run the following command.

```
tidal analyze db databases.yaml
```

2. Perform the database analysis and upload the results at a later time. The following section will explain how to run database analysis in **offline mode**

Running offline

There are circumstances in which you need to perform a database analysis on an environment without or with restricted internet access. In such case, You can perform the Database analysis, capture its results and at a later stage upload those results to your Tidal workspace.

These are the steps you need to follow in order to bypass internet access limitations:

1. On a Machine with internet access, you need to install, and configure Tidal Tools.

2. Package Tidal Tools and its required docker container images into a tar file, This will allow you to move the archive file into your restricted environment. To do so, run the following command.

   ```
   tidal backup
   ```

   Once it has finished, you will find (in your current location) a tar file called `tidal-snapshot_DATE.tar`. This is the file you need to transport into your internet restricted environment.

3. On the machine that has no internet access. You will now restore Tidal Tools using the following command.

   ```
   tidal restore tidal-snapshot_DATE.tar
   ```

   This will load a docker image and all the existing Tidal Tools configurations from the original machine. You can now run the database analysis without any external network connectivity, except to your database host itself:

   ```
   tidal analyze db --offline databases.yaml
   ```

   Note the `--offline` flag, it will indicate to Tidal Tools that the output needs to be stored in a file instead of being uploaded.

   After the analysis is completed, you will find a zip file called, `db-analyze-<DATE>.zip` that can then be transferred into a machine with internet connectivity.

4. Back to the machine with internet access. you can now upload your results to your workspace with this command.

   ```
   tidal analyze db --upload db-analyze-<DATE>.zip
   ```

   You should receive confirmation that the upload has been completed and can navigate to Tidal to see the results.

Why Docker?

You need to install Docker in order to complete the database analysis. This is because the analysis uses several system-dependent software libraries, so by using Docker the analysis can use those libraries without you requiring to install the correct dependencies with the correct versions.

What about security?

The entire analysis takes place locally on your machine. The only data that is captured and sent from the analysis are the results of the analysis and metadata. No application data, source code, files or the contents of any files on your machine are ever copied, or sent anywhere.

Advanced Configuration

Oracle Standard Edition

To use Oracle features included only in the Oracle Standard Edition (SE) license, you can set the `analyze_workload` property to `false` in your configuration file. For example:

```
databases:
  - id: 111
    name: "your-oracle-db"
    analyze_workload: false
    engine: Oracle
    host: 'my-db-host.com'
    port: 1521
    db_id_type: "Service Name"
    sid_or_service_name: "orcl"
    user: 'tidal'
    password: 'yoursecurepassword1234!'
    is_pdb: false
    cdb_sid: ""
    cbd_user: ""
    cbd_password: ""
```

{% include note.html content="By setting this to `false`, some results from the analysis will not be available, including server metrics, connected applications, and the workload analysis ranking." %}

Troubleshooting

If you are getting errors when trying to perform the analysis, it can help if you confirm that you do have network connectivity to the database.

Two commands you can use for this are:

1. `dig your_db_host` - This should return a DNS record, usually an A record, with an IP address. This means you are able to resolve the hostname of the database. If there is no IP address then you either need to adjust the hostname or you need to configure or adjust the DNS server for your operating system.

2. `nc -vzn -w 10 your_db_host db_port` This command should return `Connected to your_db_host:db_port` if it is able to connect. If it returns `Connection Timed Out` this means that it is not able to reach your database on this port. This could mean that you do not have the correct network connectivity to the database and may need to adjust firewalls or other network access. Or you are not providing the correct port that is open and listening for requests on the database.

Source code analysis

Analyzes an application's source code for various patterns that are useful in assessing applications for transformation and continual modernization.

Language support

Language	Statistics	Package Currency	IP Address Detection	Filesystem Detection	Subprocess Detection
C#	☐	☐	☐	☐	☐
Java	☐	☐	☐	☐	☐
JavaScript	☐	☐	☐	☐	☐
Python	☐	☐	☐	☐	☐
COBOL	☐	☐	☐	☐	☐
PHP	☐	☐	☐	☐	☐
C++	☐	☐	☐	☐	☐
C	☐	☐	☐	☐	☐
sh/bash/ksh/zsh	☐	☐	☐	☐	☐
JSP	☐	☐	☐	☐	☐
Perl	☐	☐	☐	☐	☐
Visual Basic, VB.NET, VBScript	☐	☐	☐	☐	☐

It is also able to determine statistics and IP address detection for almost any source code files including but not limited to:

Ada, ASP, ASP.NET, Assembly, AWK, Basic, Batch, C Header, C Shell, C++ Header, Clojure, ClojureScript, CMake, CoffeeScript, CSS, Cython, Coldfusion, D, Dart, Dockerfile, Elixir, Elm, Emacs Lisp, Erlang, F#, Fish, modern and legacy Fortran, Groovy, Go, Handlebars, Haskell, HTML, INI, Jupyter, Kotlin, LaTeX, Less, Lisp, Lua, Makefile, Matlab, Nushell, Objective-C, Objective C++, OCaml, Pascal, PL/SQL, Powershell, R, Rexx, Ruby, Rust, Sass, Scala, Scheme, sed, Swift, Terraform, and TypeScript.

What does it do

It measures and determines several important aspects of an application code base including;

- Statistics - Overall code base statistics, including statistics regarding the total number of files, lines of code, comments and blank lines, types of languages used, as well as these statistics per source file.
- Package Currency - Library and package detection, including the names and versions of all packages used.
- Filesystem Detection - Filesystem usage, reading, writing, or manipulation of files stored on the OS filesystem.
- Subprocess Detection - Usage of executing subprocesses outside of the execution environment of the language.

35

- IP Address Detection - Identification of references to IP addresses, private addresses, and addresses used as listeners.

What information is collected

The entire code analysis process takes place locally on your machine. It will scan your files locally looking for common patterns and information when modernizing.

This information includes:

- Statistics of total number of lines, by language. For example:

```
{"Blank": 575,
 "Bytes": 95722,
 "Code": 2530,
 "CodeBytes": 0,
 "Comment": 299,
 "Complexity": 183,
 "Count": 42,
 "Files": [],
 "Lines": 3404,
 "Name": "JavaScript",
 "WeightedComplexity": 0}
```

- Filepaths of files analyzed, for example:

```
"/src/home/pnr/tidal/schoolbus/Server/SchoolBusAPI/Authentication/SbJwtBearerEvents.cs",
"/src/home/pnr/tidal/schoolbus/Server/SchoolBusAPI/Authorization/ClaimsPrincipalExtensions.cs",
"/src/home/pnr/tidal/schoolbus/Server/SchoolBusAPI/Authorization/PermissionHandler.cs",
```

- Identified software dependencies package names, and versions, for example:

```
{
  "from_file": "/src/home/pnr/tidal/schoolbus/Server/Serilog.Sinks.SbiPostgreSql/Serilog.Sinks.SbiPo
  "language": "CSharp",
  "name": ".NET Core",
  "owner": null,
  "version": "3.1"
},
{
  "from_file": "/src/home/pnr/tidal/schoolbus/client/package-lock.json",
  "language": "JavaScript",
  "name": "@babel/code-frame",
  "owner": null,
  "version": "7.10.4"
}
```

- Identified patterns and implementations in code, for example, the detection of filesystem usage in Python:

```
{
  "check_id": "filesystem-python",
  "end": {
    "col": 19,
    "line": 132,
    "offset": 3703
  },
```

36

```json
  "extra": {
    "engine_kind": "OSS",
    "fingerprint": "f50c74db097532ad463e76e178545defc5ed47fbaf364527da876a75e33481708f97308217f2364b
    "is_ignored": false,
    "lines": "    if not os.path.exists(output_dir):",
    "message": "Filesystem usage",
    "metadata": {},
    "metavars": {},
    "severity": "INFO"
  },
  "path": "/src/home/pnr/tidal/schoolbus/ApiSpec/CCWJurisdiction/csv2json.py",
  "start": {
    "col": 12,
    "line": 132,
    "offset": 3696
  }
}
```

These are snippets from a real analysis results file. To verify and confirm these findings, and see what the full results include on either your application or a sample application see the next section below. ## Validate and confirm You can view all of the raw analysis results before they are sent or seen by anyone. You can perform the entire analysis locally without any network connectivity to ensure this. The only data that is captured and sent from the analysis are the results of the analysis and the associated metadata.

To see an example of the information that is captured and sent:

1. Download a copy of this open-source 'schoolbus' application[39] by cloning it with `git`[40] or downloading a zip file[41] (and then unzipping it) from GitHub. If you have source code locally that you would like to try it on we recommend you use that.

2. Move to the schoolbus directory, download the needed docker images, and run the code analysis locally:

```
cd schoolbus
tidal backup
tidal analyze code --offline
```

The `--offline` flag indicates that the results are saved to a local file and no data is retrieved or sent on any network. This is why the first `backup` command is needed, to download the necessary analysis components. Only the `backup` command requires internet access. The `--offline` flag by design does not use any network connectivity, and if you want you can disable network connectivity before running the analyze command.

3. After the analysis is completed, open the file (using any text editor) that was created in your current directory, `code-analysis-<DATE>-all.json`. You can review all of the information that is included in this results file. Only the information in this file is sent, processed, and stored by Tidal. You can see here a complete results file from an analysis of the 'schoolbus' application.

4. If you have any questions about the contents of the results please let us know. Once you are satisfied with the contents of the results you can send the results to your Tidal Workspace:

```
tidal analyze code upload --app-id 1 ./path-to-results-file
```

How to use it

For step-by-step instructions to perform an analysis refer to the usage guide.

---

[39] https://github.com/tidalmigrations/schoolbus
[40] https://help.github.com/en/github/creating-cloning-and-archiving-repositories/cloning-a-repository
[41] https://docs.github.com/en/repositories/working-with-files/using-files/downloading-source-code-archives#downloading-source-code-archives-from-the-repository-view

Tidal Source Code Analysis

Not sure how ready you are to move to the cloud? With Tidal you can analyze your application's source code to identify difficulties and opportunities in migrating and modernizing. It is compatible with most of the current and legacy languages and detects a wide array of factors. See the overview for more information on its capabilities.

{% include image.html file="source_code_analyze.png" caption="Analyze your source code" %}

What You Need to Get Started

1. Install, and authenticate Tidal Tools with your Tidal Accelerator Workspace.

2. You will need the ID of the application you are going to perform the source code analysis on. You can find it in the URL bar when looking at an application in Accelerator. ex. If you are viewing an application in Tidal, the URL will show `https://[your workspace].tidal.cloud/apps/111/overview` in this case 111 is the application ID.

3. You will need a copy of the source code files for the application locally on your device.

With that, you can perform the code analysis in a matter of seconds.

{% include tip.html content="Looking to try it out and don't have any code handy? You can use this sample 'schoolbus' application[42] by cloning it from GitHub[43]." %}

Performing the Code Analysis

When analyzing an application there are two workflows available. Deciding which option to use depends if the code you are analyzing is located on a machine that has access to the internet or not, specifically https://tidal.cloud. If you do have the code on a machine with this access you can use the standard mode. If the code is located on a device that can not access the Accelerator platform, then you can use the Offline mode.

Standard mode

1. Perform the source code analysis and upload the result immediately to your workspace:

   ```
   cd /path/to/source-code
   tidal analyze code --app-id [app_id_for_your_application]
   ```

   Alternatively, you can pass multiple locations or even wildcard paths, or filenames for analysis. This will result in all the provided and matching file paths to be considered source code for the one application you are analyzing.

   ```
   tidal analyze code [/path/to/source-code-A] [/path/to/source-code-B] --app-id [app_id_for_your_a
   ```

Offline mode

If the machine running code analysis can not access the internet; you can perform the analysis and store the results locally and upload those results to your workspace from a different machine. This process will involve two machines, one that has internet access and one that has no internet access, and a copy of the application's source code.

---

[42]https://github.com/tidalmigrations/schoolbus
[43]https://help.github.com/en/github/creating-cloning-and-archiving-repositories/cloning-a-repository

1. On a machine with internet access, install, and configure Tidal Tools.

2. Backup Tidal Tools required dependencies into a backup file.

   `tidal backup`

   When finished in the current working directory there will be a file called `tidal-snapshot_DATE.tar`. Move this file, in any reasonable way you can, to the machine with the code on it to be analyzed

3. On the machine that has no internet access, you will now restore the backup:

   `tidal restore tidal-snapshot_DATE.tar`

   This will load the needed dependencies and Tidal Tools configuration needed to run the analysis.

4. Now run the source code analysis on the current directory:

   ```
   cd /path/to/source-code
   tidal analyze code --offline
       ```
   ```

   ```
   Alternatively, you can pass multiple locations or even wildcard paths, or filenames for analysis
   ```

   ```bash
   tidal analyze code [/path/to/source-code-A] [/path/to/source-code-B] --offline
   ```

   The `--offline` flag indicates that the results are saved to a local file, instead of being uploaded.

   After the analysis is completed, you will find the results in a file called `code-analysis-<DATE>-<TYPE>.json`.

5. Transfer the results file from the analysis to the machine with internet access.

6. On the machine with internet access, upload the results to your workspace:

   ```
   tidal analyze code upload [file_name] --app-id [app_id_for_your_application]`
   ```

   You should receive confirmation that the upload has been completed and can navigate to Tidal Accelerator to see the results.

{% include tip.html content="Need to run code analysis on an entire set of applications all from one machine? Run this command and easily create a directory for every application already in your Tidal Accelerator workspace[44]." %}

Why do I need Docker?

You need to install Docker to complete the source code analysis. This is because the analysis uses several system-dependent software libraries. By using Docker, the analysis can use those libraries without requiring you to install any other dependencies.

What information is collected?

The entire code analysis process takes place locally on your machine. It will scan your files locally looking for common patterns and information when modernizing. The only data that is captured and sent from the analysis are the results of the analysis and the associated metadata. You can view all raw analysis results sent to Tidal Accelerator before they are sent, by performing an offline analysis and viewing the results file locally. See the overview for more information on how to do that.

---

[44]https://github.com/tidalmigrations/gists/blob/master/make_source_code_dirs.sh

Summary

Congratulations! You have completed your discovery across your portfolio.

We hope that this book helped you to get yourself familiar with **Layering Discovery Techniques**:

1. Importing spreadsheets of applications, databases, servers, and virtualization clusters
2. Scheduling synchronizations across your hypervisors such as VMWare or Hyper-V
3. Server usage statistics aggregation
4. Discovering and fingerprinting your running web applications
5. Performing databases analysis
6. Analyzing your source code

That was the beginning of your journey towards the transformative cloud migrations. And now, we hope, you are ready to move forward!

Next Steps

Now that you have completed your discovery across your portfolio, you are ready to do your assessment. The first step is to make sure you have a set of discrete applications in scope for your migration project. Navigating to your list of `Applications` under the `ASSESS` section in the platform, you should be able to see a list of all your applications. From here, you can tag them with a specific tag for your project, and even create a dashboard tile that includes all the items in scope.

With the list of applications and a full set of data from the discovery process, you can now perform an initial assessment and determine a transition strategy (one of the 6 Rs) for each application.

As well with this data, you can generate a report, from the `Portfolio Reports` section under `PLAN`, in order to get a full report on your assessment. This report can be shared with other parties and people interested in the project progress.

Appendix

Technical troubleshooting

Most problems with Tidal Tools can be fixed by following the troubleshooting methods described below. If you need extra help with any of this, you can reach us at support@tidalcloud.com.

General troubleshooting options

Diagnose dependencies and environment with `tidal doctor`

To work properly and to provide some of its features Tidal Tools depends on some additional software and external services. To check your environment run `tidal doctor` and review its output for any warnings and ways to recover.

Sample `tidal doctor` output looks like the following:

```
[-] Tidal Tools v2.2.20
    X Updates available.
      Go to https://get.tidal.sh/ for update instructions.
    • Config file in use: /home/tidaluser/.config/tidal/config.yaml
    • Logfile location: /home/tidaluser/.local/share/tidal/tidal.log

[√] Tidal API connection
    • Tidal API connection configured OK.

[√] Docker
    • Docker at /usr/bin/docker
    • Server 18.09.0 • API 1.39 (min. 1.12) • Client 1.39
    • Pull from registry OK

[-] DNS Tools
    X DNS Tools not installed.
      Go to https://dnstools.ninja/download/ for installation instructions

[-] vSphere connection
    X vSphere connection not configured.
      Run 'tidal login vsphere' to set it up.
```

Update Tidal Tools

Periodically Tidal Tools checks if the newer version is available. It would inform you about it as a message printed to your terminal or command prompt output after any `tidal` command invocation:

```
Looks like you are running an older version of Tidal Tools.
Check https://get.tidal.sh for update instructions!
```

Also you can explicitly check for new Tidal Tools versions available by running `tidal check-updates` or `tidal doctor` command.

Default log file location

To perform better diagnosis of any issues with Tidal Tools, some of the underlying activities performed by Tidal Tools are written to the log file. If you are having trouble with one of the Tidal Tools commands, you can reach us at support@tidalcloud.com and send us your log file for us to investigate.

The default locations of the log file are:

- `C:\Users\tidaluser\AppData\Roaming\tidal\tidal.log` on *Windows*
- `/home/tidaluser/.local/share/tidal/tidal.log` on *Linux*
- `/Users/tidaluser/Library/tidal/tidal.log` on *macOS*

Please note that log files are truncated by default before the `tidal` invocation. That means that the default log file contains entries specific to the one particular `tidal` run.

How to prevent log file from truncating

The default log files truncation behavior may be not desirable in cases when it is necessary to combine log entries of the several subsequent `tidal` invocations. To prevent the existing log file from truncating you can utilize `--keep-log` command line flag. When the `--keep-log` flag is used the log file (default, or one specified with `--log-file`) won't be truncated before the `tidal` command call.

Docker installation

Some of the Tidal Tools features (for example, `tidal analyze code`) depend on Docker to be installed. To get the Docker installation instructions please check the Docker Documentation[45].

How to check if Docker actually works

There are a few ways to check if your Docker installation actually works.

Open your terminal emulator or command prompt and run the command `docker run hello-world`. The sample output should look like the following:

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
d1725b59e92d: Pull complete
Digest: sha256:0add3ace90ecb4adbf7777e9aacf18357296e799f81cabc9fde470971e499788
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
…
```

To explicitly check that your Docker installation is able to communicate with Tidal container registry run the command `docker run gcr.io/tidal-1529434400027/hello-world`. You should see the output similar to the following:

---

[45]https://docs.docker.com/install/

```
Unable to find image 'gcr.io/tidal-1529434400027/hello-world:latest' locally
latest: Pulling from tidal-1529434400027/hello-world
9e91b00c0251: Pull complete
Digest: sha256:b60c2de90e0b5c4f7e74b84ca888e2fec1d288d47c99e48bd612e0eefeb604c5
Status: Downloaded newer image for gcr.io/tidal-1529434400027/hello-world:latest
Hello from Tidal!
```

Another option is to run `tidal doctor` and check the Docker section of the output.

Docker networking issues

Sometimes Docker has issues with DNS[46] and you may see an error like:

```
Error response from daemon: Get https://gcr.io/v2/: proxyconnect tcp: dial tcp: lookup http on 192.1
```

If so you can update the DNS server used by docker[47] to `8.8.8.8`, as Docker recommends, to solve the issue.

Specifying Docker Proxy

If you need a proxy to access the internet and see an error that looks similar this:

```
Error response from daemon: Get https://gcr.io/v2/: net/http: request canceled while waiting for con
```

You may need to configure Docker to use your proxy server. You can follow the steps that Docker provides to do that on Windows[48] or Linux[49], depending on where you are running docker.

{% include note.html content="If you need to authenticate with the proxy, be sure to include the username and password in the value, ie. 'http://proxy_userid:proxy_password@proxy_ip:proxy_port'" %}

PostgreSQL database fails to analyze with `tidal analyze db`

Check the PostgreSQL Server User[50] script to verify that the all the necessary permissions were granted. Pay attention that different permissions should be applied to different PostgreSQL versions.

Windows specific troubleshooting

Setting up Docker for Windows to use Linux containers

Docker for Windows supports both Linux containers and Windows containers, however Tidal Tools works when your Docker installation is set to to use Linux containers.

To check if your Docker installation was configured to use Linux containers please check one of the following:

- Open Docker for Windows menu and check for the item **Switch to Windows containers**. If you can find it, that means that your Docker is configured to talk with Linux daemon, so no further actions are needed to be performed.

---

[46]https://docs.docker.com/docker-for-windows/troubleshoot/#networking-issues
[47]https://docs.docker.com/docker-for-windows/troubleshoot/#networking-issues
[48]https://docs.docker.com/docker-for-windows/#proxies
[49]https://docs.docker.com/network/proxy/
[50]https://guides.tidal.cloud/analyze-database.html#postgresql-server-user

- However, if you see **Switch to Linux containers** in the Docker for Windows menu, that means that the Docker was configured to talk to Windows daemon. You need to click that menu item to switch to Linux containers.
- Running `tidal doctor` also checks if your Docker installation is configured to use Linux containers or not.

Issues running Tidal Tools with PowerShell ISE

Since PowerShell ISE only runs console apps that don't require user input some of the Tidal Tools commands may work incorrect. It's recommended to use PowerShell instead (i.e. `PowerShell.exe`, **not** `PowerShell_ise.exe`).

However if you really need to use PowerShell ISE you should make some preparations. Most of the time Tidal Tools ask user for some input is when it prompts for connection credentials. To prevent Tidal Tools from prompting you should explicitly specify all the necessary connection information. It could be done by either using `tidal config` command, or manually editing the configuration file, or by setting up the appropriate environment variables.

Windows directory separators in YAML files

Windows traditionally uses the backslash (`\`) to separate directories in file paths. For example, `C:\Program Files\Tidal Software\tidal`. However, the configuration and discovery plan language (YAML[51]) also uses the backslash (`\`) as an escape character in quoted strings[52]. This can make it awkward to write literal backslashes.

Generally it is OK to use forward slashes, because most of the time the Windows file system APIs will accept **both** the backslash (`\`) and forward-slash (`/`) in file paths. But when you use backslashes, you must pay extra attention to keep them from being suppressed by YAML string quoting. That means that you must escape the backslash character with another backslash (which is the escape character), so the string must be as the following: `C:\\Program Files\\Tidal Software\\tidal`.

Linux troubleshooting

Manage Docker as a non-root user

The Docker daemon binds to a Unix socket instead of a TCP port. By default that Unix socket is owned by the user `root` and other users can only access it using `sudo`. The Docker daemon always runs as the `root` user.

If you don't want to preface the `tidal analyze code` command with `sudo`, create a Unix group called `docker` and add users to it. When the Docker daemon starts, it creates a Unix socket accessible by members of the `docker` group.

To create the `docker` group and add your user:

1. Create the `docker` group.

```
$ sudo groupadd docker
```

2. Add your user to the `docker` group.

```
$ sudo usermod -aG docker $USER
```

---

[51] https://yaml.org/
[52] https://yaml.org/spec/1.2/spec.html#id2776092

3. Log out and log back in so that your group membership is re-evaluated. If testing on a virtual machine, it may be necessary to restart the virtual machine for changes to take effect. On a desktop Linux environment such as X Windows, log out of your session completely and then log back in.

4. Verify that you can run `docker` commands without `sudo`.

```
$ docker run hello-world
```

"docker: Error response from daemon: OCI runtime create failed" error message on Fedora 31

Fedora 31 is the first major Linux dustribution that comes with cgroup v2 enabled by default. However, Docker still do not support cgroup v2.

To start Docker on Fedora 31 run the following command and reboot:

```
$ sudo dnf install -y grubby && \
  sudo grubby \
  --update-kernel=ALL \
  --args="systemd.unified_cgroup_hierarchy=0"
```

This command reverts the systemd configuration to use cgroup v1.