ORACLE
Cloud Infrastructure

Resource Manager

Level 100

Flavio Pereira

February 2019

ORACLE

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

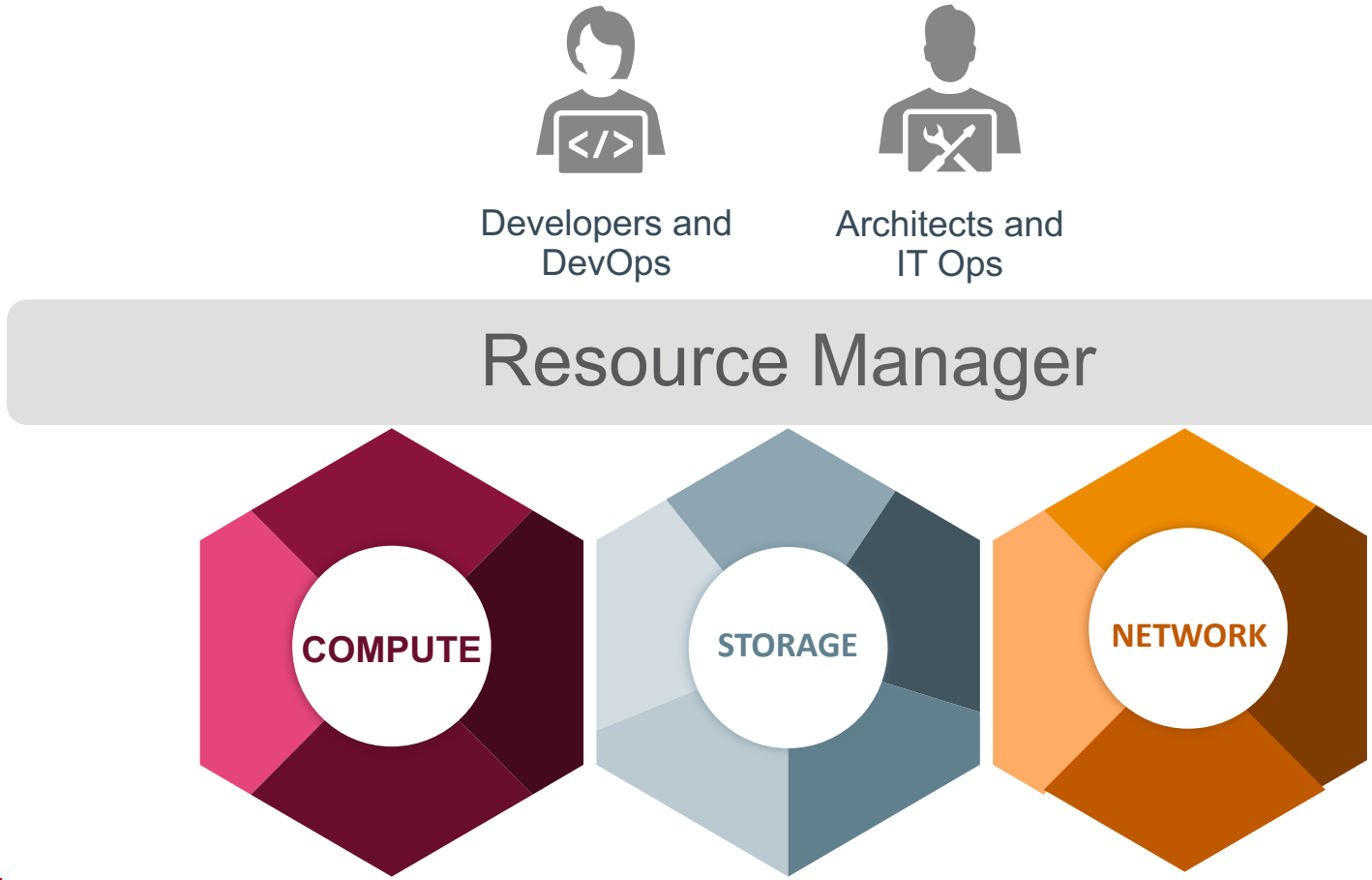
Objectives

After completing this lesson, you should be able to:

- Describe the basic components of Resource Manager
- Describe the benefits of Resource Manager
- Prepare Terraform files for Resource Manager
- Resource Manager Demo

Introducing the Oracle Cloud Infrastructure Resource Manager

Manage your infrastructure resources using Terraform



Resource Manager Benefits

- Automate and standardize your infrastructure and easily replicate environments
- Deep integration with OCI Platform and its services
- Seamlessly manages state files and improves team collaboration
- Fully managed service for the Terraform engine
- You only pay for the underlying compute, storage, network or any other resource you provision using Resource Manager.

Resource Manager Components

You can think of the Resource Manager as Terraform-as-a-Service for Oracle Cloud Infrastructure resources. Once you have your terraform files ready and your variables file adjusted for Resource Manager you can start building **Stacks** and executing **Jobs:**

Stack - Represents a set of OCI resources you want to create in a compartment. Each Stack has a *configuration*, which is a set of Terraform files that specify the resources you want to manage together using the Resource Manager.

Job - Represents a request to take a Terraform Action on a Stack. Resource Manager provides three job-types: **plan, apply and destroy**

Resource Manager - Access Control & Permissions

- To use the Resource Manager, you must have the required OCI Identity and Access Management (IAM) permissions.

- Following are example policy statements that grant a group called ADMIN-XYZ the ability to manage all the Resource Manager resources in the tenancy:

```
Allow group ADMIN-XYZ to manage orm-stacks in tenancy
Allow group ADMIN-XYZ to manage orm-jobs in tenancy
```

- DEV-XYZ is a developer group is able to read Stacks and to execute Jobs - except for destroy – only within in a specific compartment:

```
Allow group DEV-XYZ to use orm-stacks in compartment XYZ
Allow group DEV-XYZ to use orm-jobs in compartment XYZ where target.job.operation !=
'DESTROY'
```

Prepare your Terraform files to work with Resource Manager

- With the Resource Manager all that is required is an OCI Identity and Access Management (IAM) permissions.

- You can omit the user OCID, private key, fingerprint, and tenancy OCID from provider configuration.

Open Source OCI Terraform Provider

```
variable "user_ocid" {}
variable "fingerprint" {}
variable "private_key_path" {}
variable "private_key_password" {}

provider "oci" {
  tenancy_ocid         = "${var.tenancy_ocid}"
  user_ocid            = "${var.user_ocid}"
  fingerprint          = "${var.fingerprint}"
  private_key_path     = "${var.private_key_path}"
  private_key_password = "${var.private_key_password}"
  region               = "${var.region}"
}
```

Resource Manager

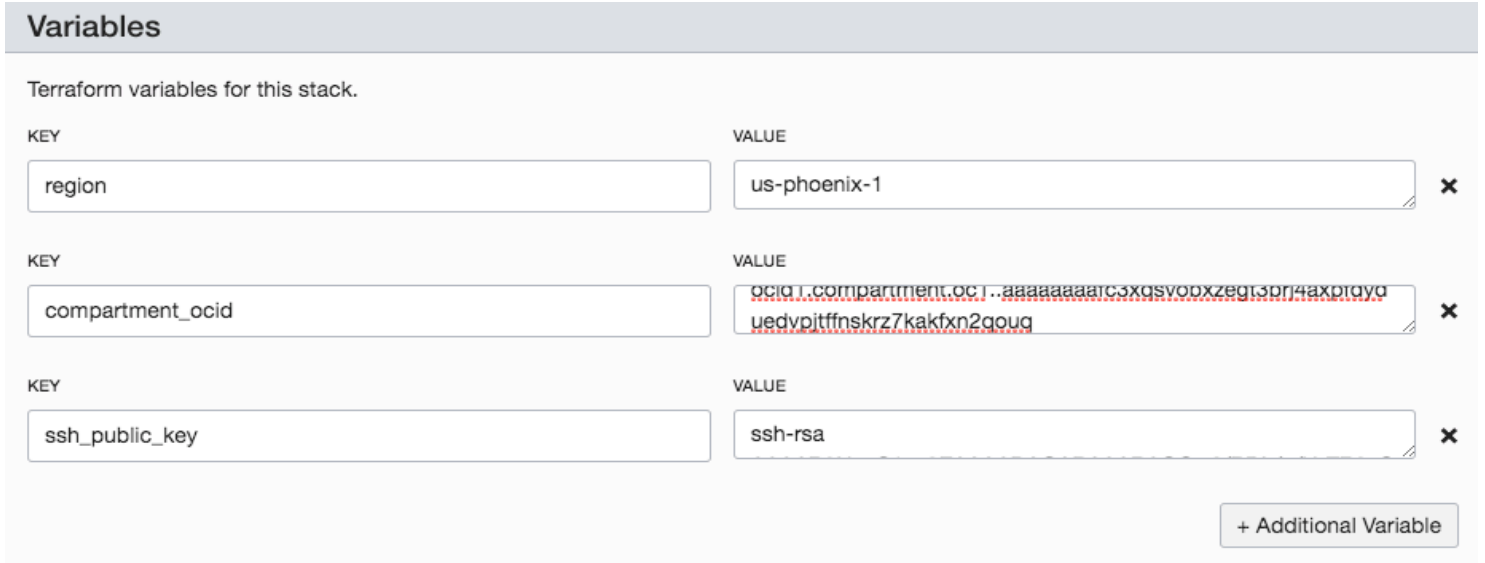
```
variable "region" {}
variable "compartment_ocid" {}

provider "oci" {
  region = "${var.region}"
}
```

Resource Manager Variables

Resource Manager give you a possibility to enter extra variables to help with your deployment. Here is an example of adding a public ssh key:

Terraform Resource Manager file

```
variable "region" {}
variable "compartment_ocid" {}
variable “ssh_public_key” {}
```



Resource Manager Workflow: Step 1

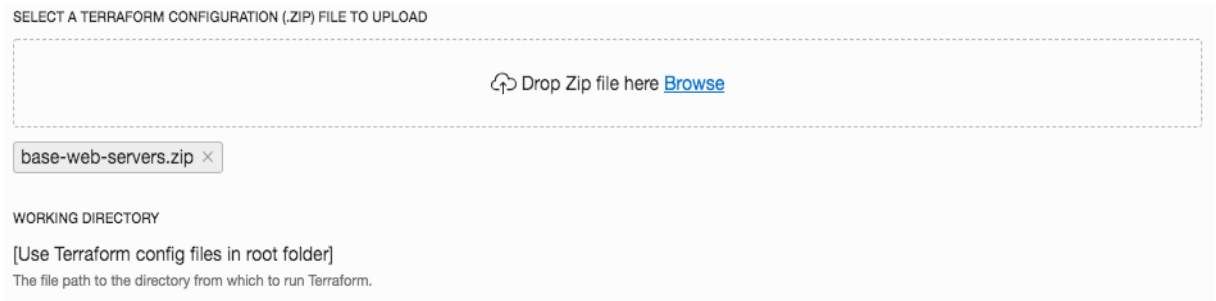
Define configuration

Terraform

Define configuration

Create a Stack

Run a Job



- Create the Terraform configuration for resources
- Write optional Terraform modules
- Create a zip file containing the Terraform files
- Avoid supplying confidential information in your configuration like passwords and SSH keys.

Resource Manager Workflow: Step 2

Create a Stack

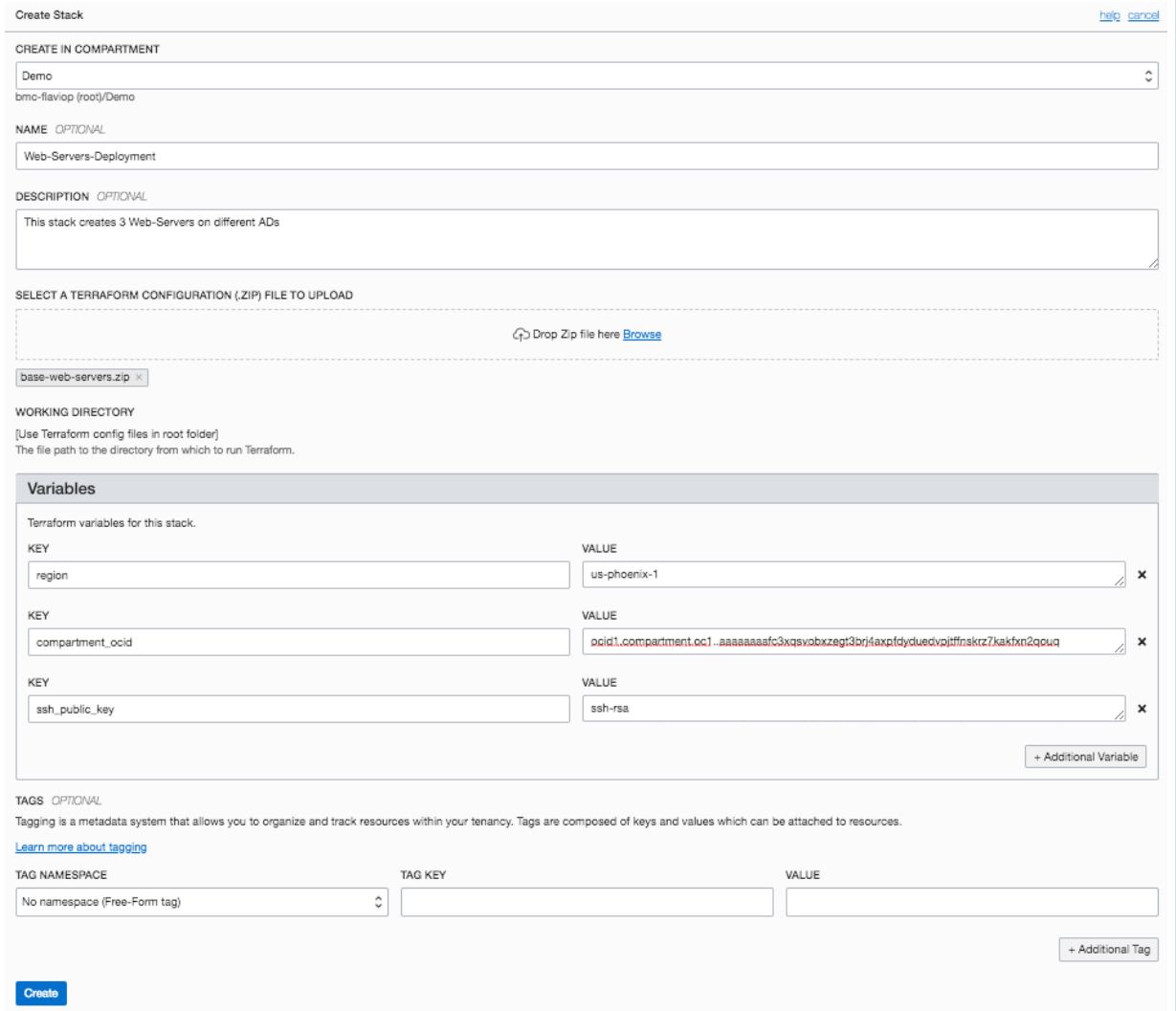
Terraform

Define configuration

Create a Stack

Run a Job

- Stack represents a set of resources you manage within a compartment
- Each Stack maps to Terraform configuration files and a Terraform state file



Resource Manager Workflow: Step 3

Run a Terraform Job

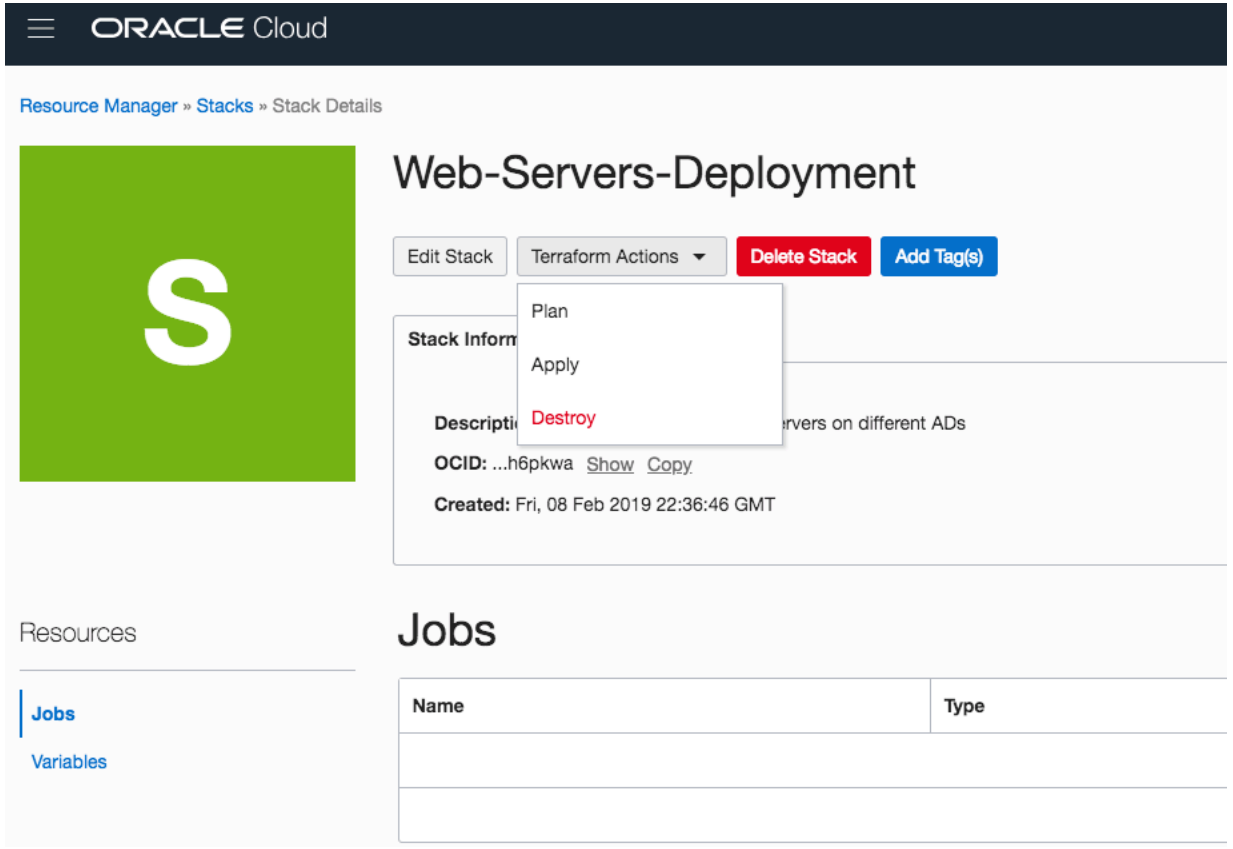
Terraform

Define configuration

Create a Stack

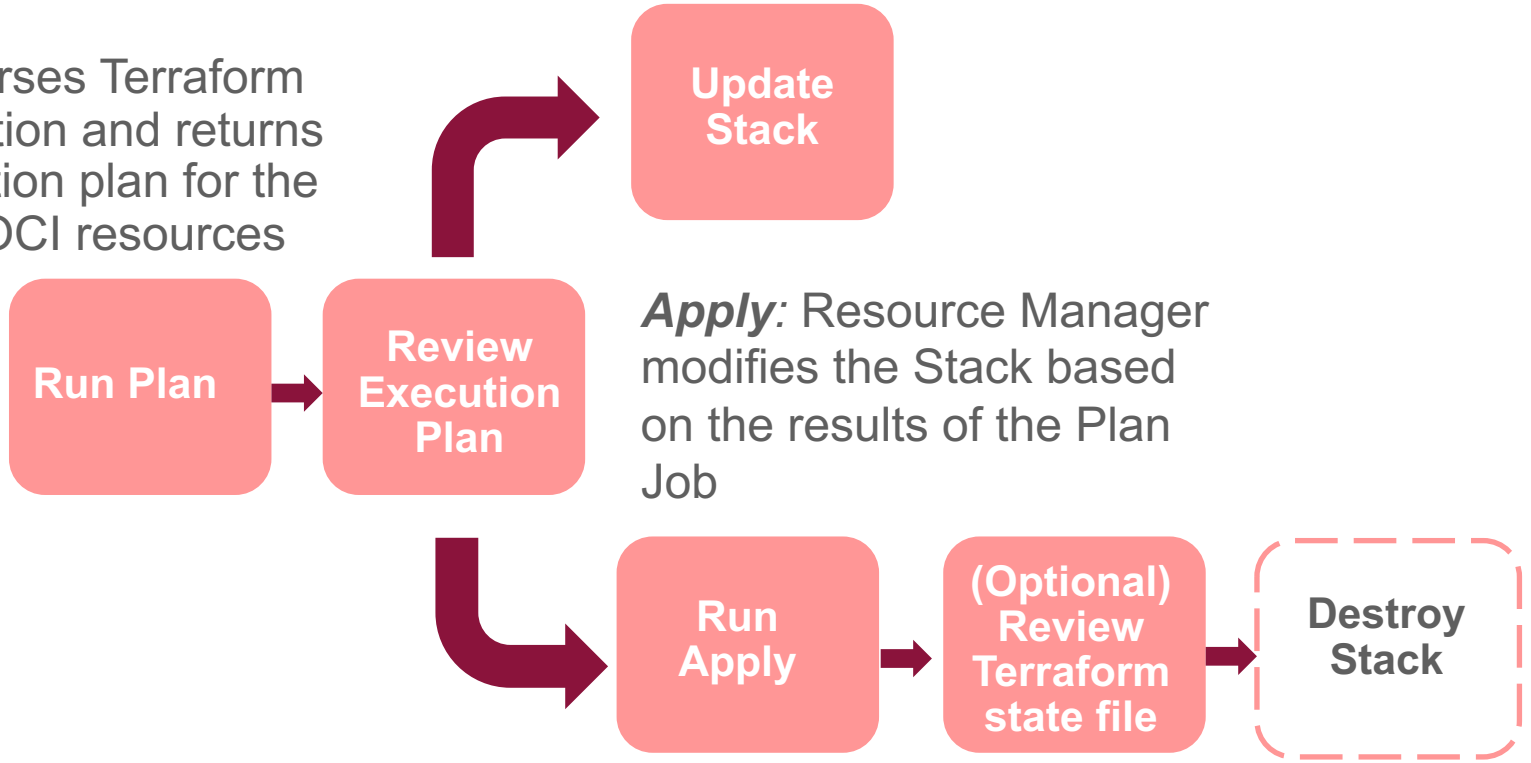
Run a Job

- A Job is a Terraform Action executed against a Stack
- Job actions include Plan, Apply, and Destroy



Resource Manager Execution

Plan : Parses Terraform configuration and returns an execution plan for the effected OCI resources



Apply*:* Resource Manager modifies the Stack based on the results of the Plan Job

Destroy :Terraform deprovisions all of the resources in the Stack

Resource Manager Demo

Summary

- Build on Open Source Software, Resource Manager is fully-managed service that makes easier to use Terraform on Oracle Cloud Infrastructure
- You can leverage your existing Terraform templates to deploy with Resource Manager
- There are no charges for using the Oracle Cloud Infrastructure Resource Manager.
- You can try Resource Manager, by sign up for a free trial OCI account here: https://cloud.oracle.com/tryit and follow the steps on this guide: http://bit.ly/hol-orm
- Resource Manager Documentation: https://docs.cloud.oracle.com/iaas/Content/ResourceManager/Concepts/resourcemanager.htm

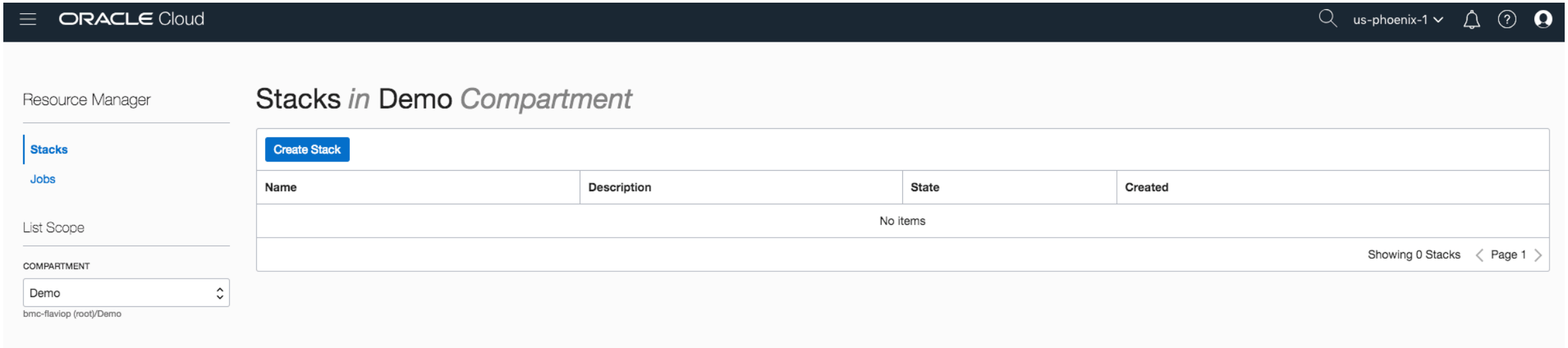
ORACLE®
Cloud Infrastructure

cloud.oracle.com/iaas

cloud.oracle.com/tryit

Resource Manager Demo: Step 1

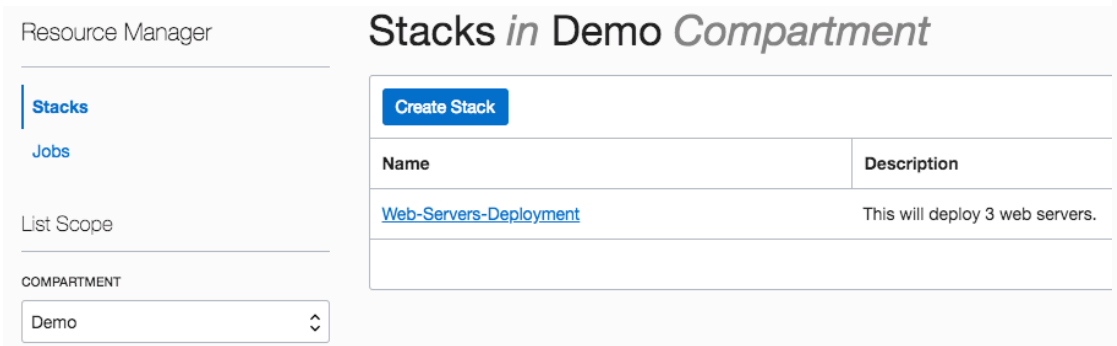
Go to **Menu** → **Resource Manager** → **Stack** and click **Create Stack**

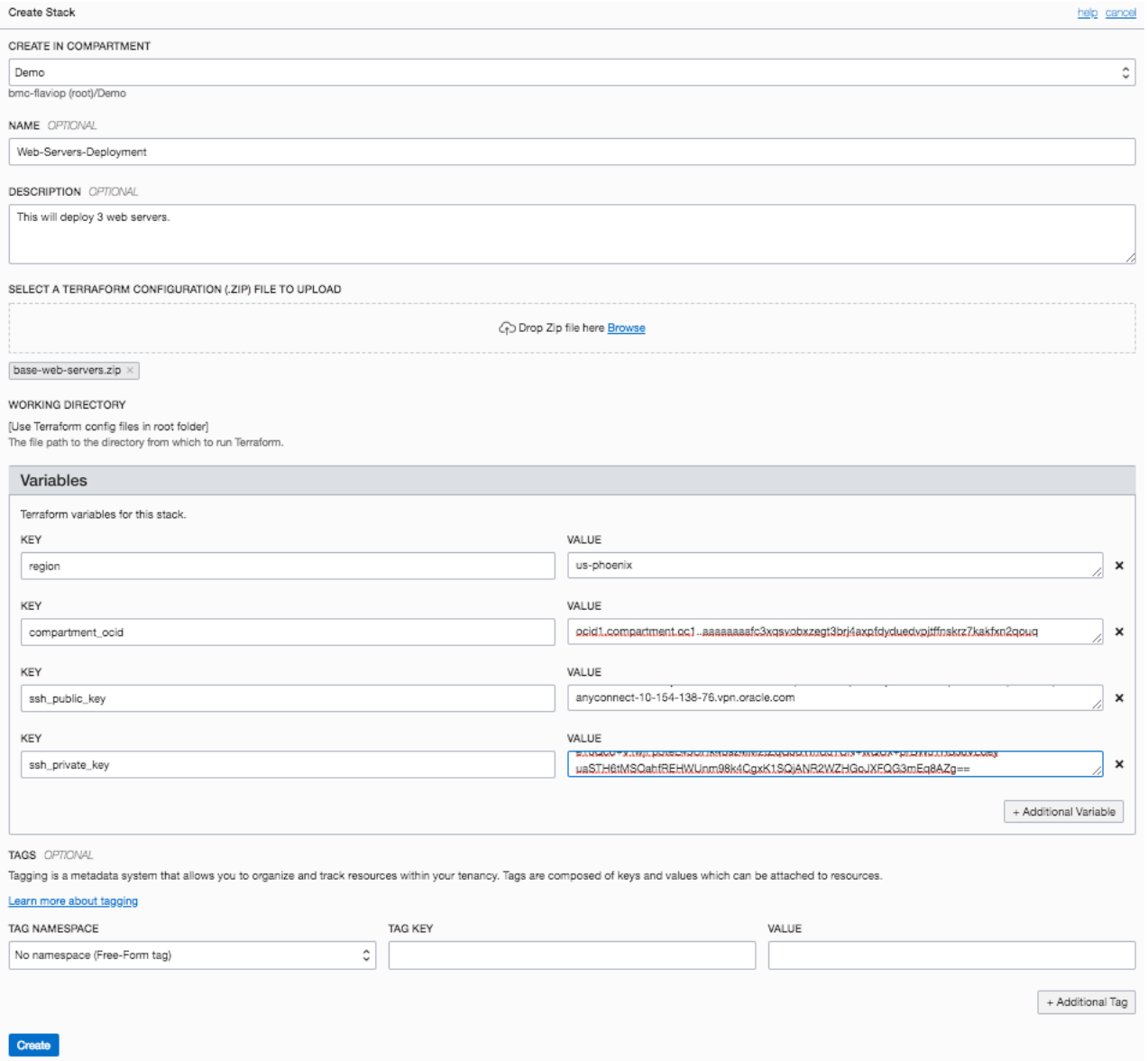


Resource Manager Demo: Step 2

- Select the target compartment
- Enter a Name and Description
- Upload your Terraform zip file
- Enter the variables
- Click Create

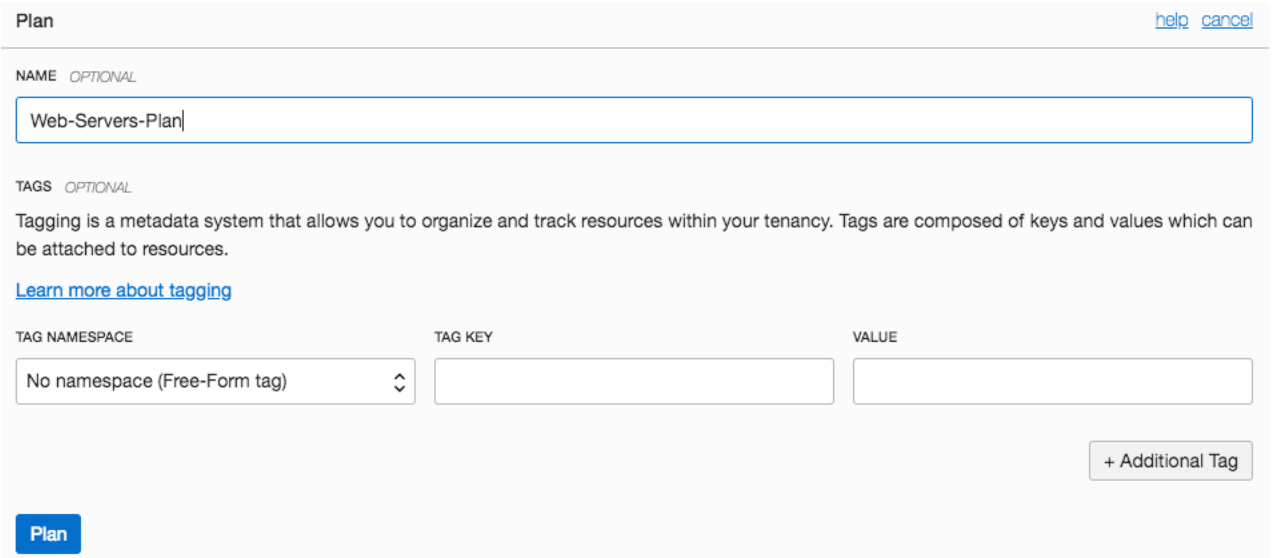
You will see your Stack created showing an **Active State**

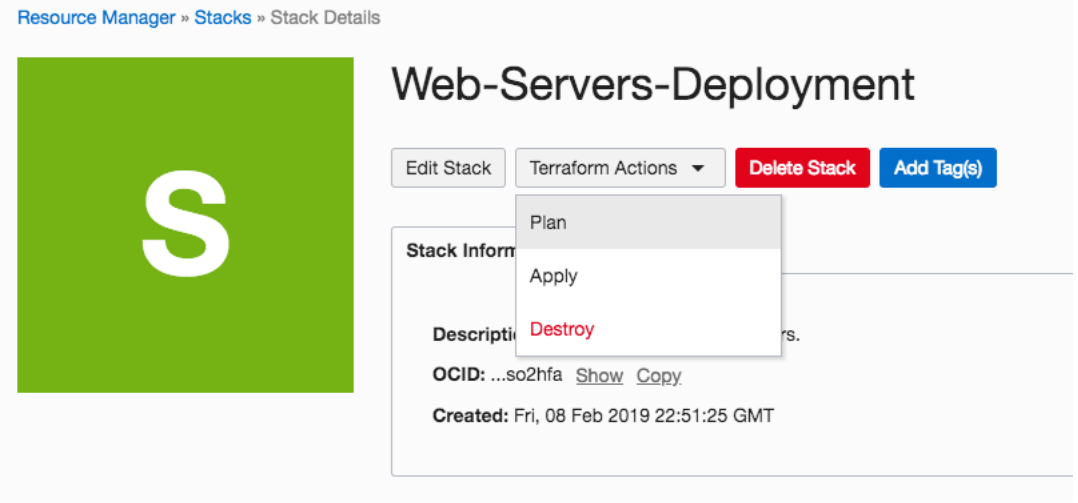




Resource Manager Demo: Step 3

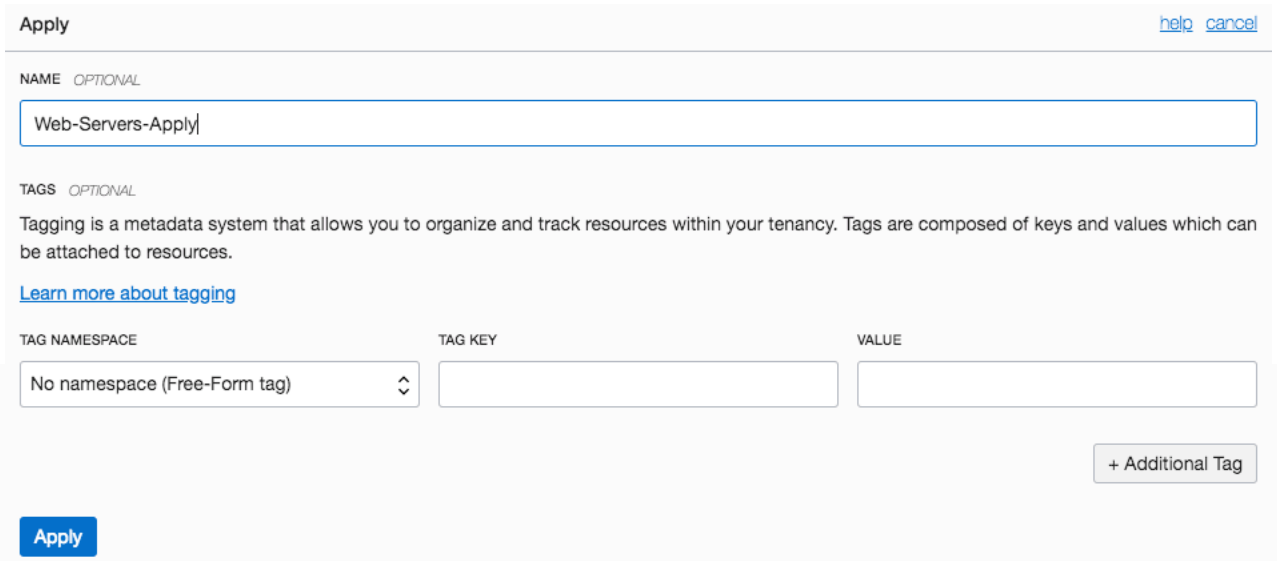
- Select the Stack you created
- Select ”Plan” on the Terraform Actions menu
- Give it a name and click the “Plan” button

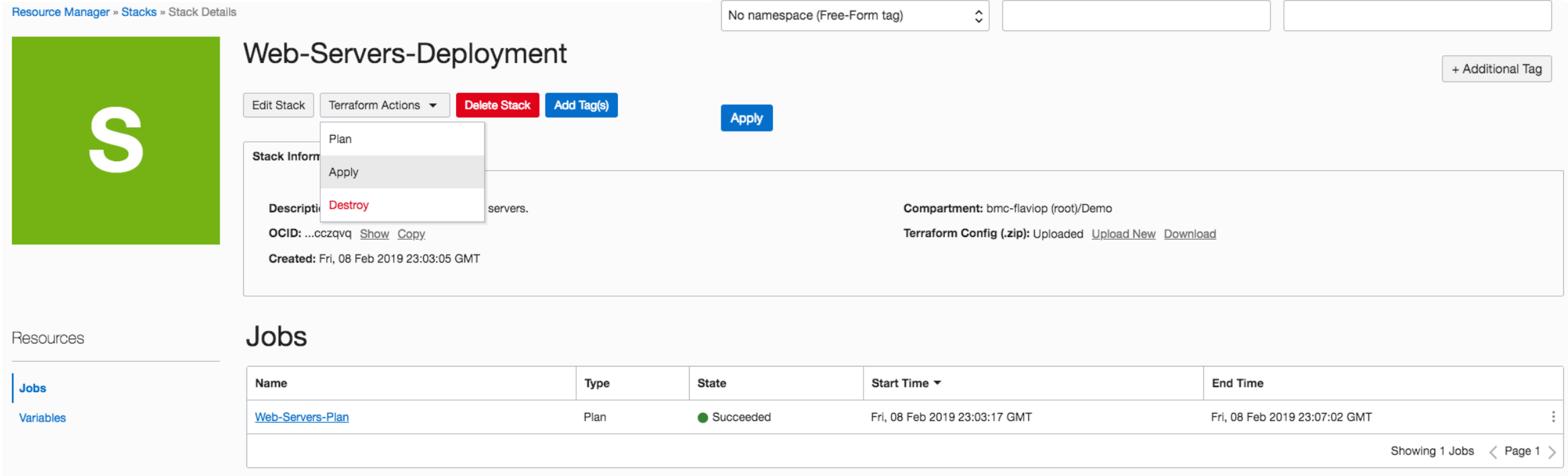




Resource Manager Demo: Step 4

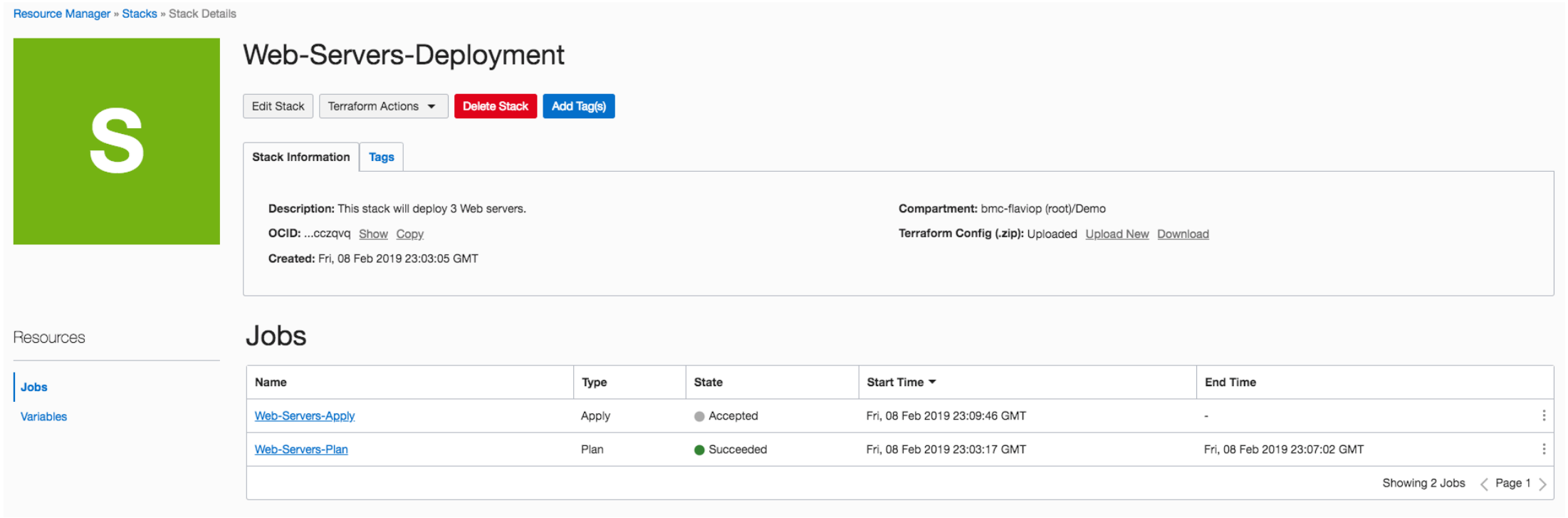
- Select ”Apply” on the Terraform Actions menu
- Give it a name and click the “Apply” button





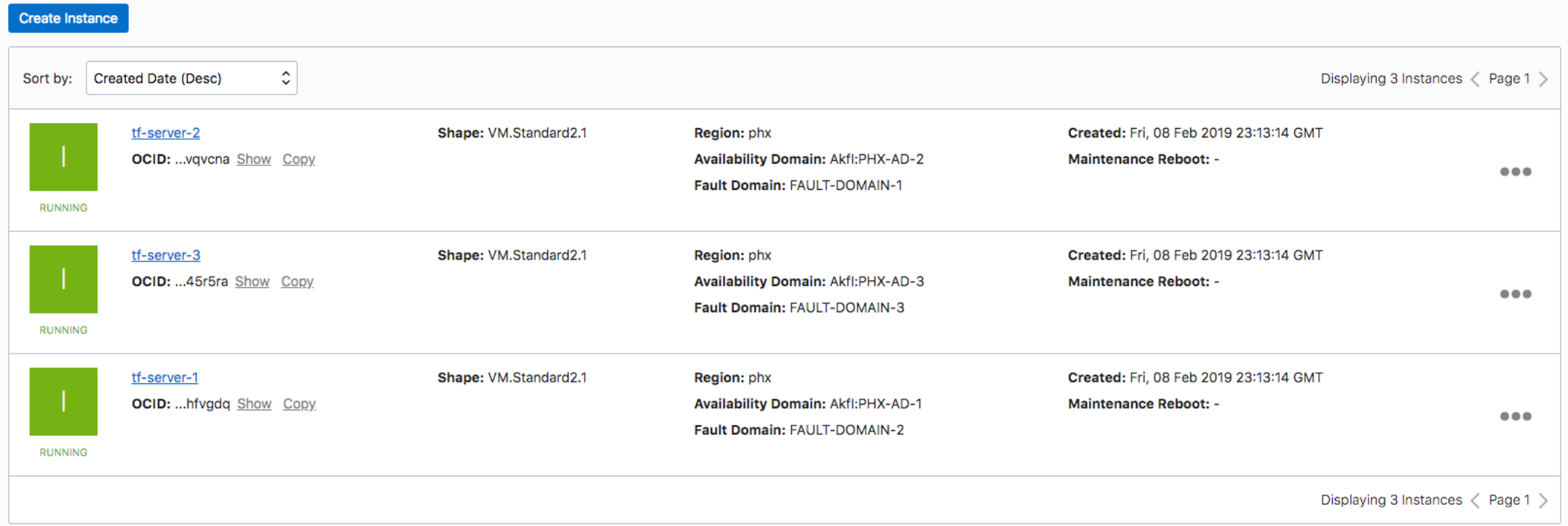
Resource Manager Demo: Step 5

Under Jobs you can see the job history and the state of the actions.



Resource Manager Demo: Step 6

Navigate to **Compute** > **Instances** and see the created Web-Servers



Resource Manager Demo: Step 7

For each Job, download the Logs, Terraform configuration and Terraform State