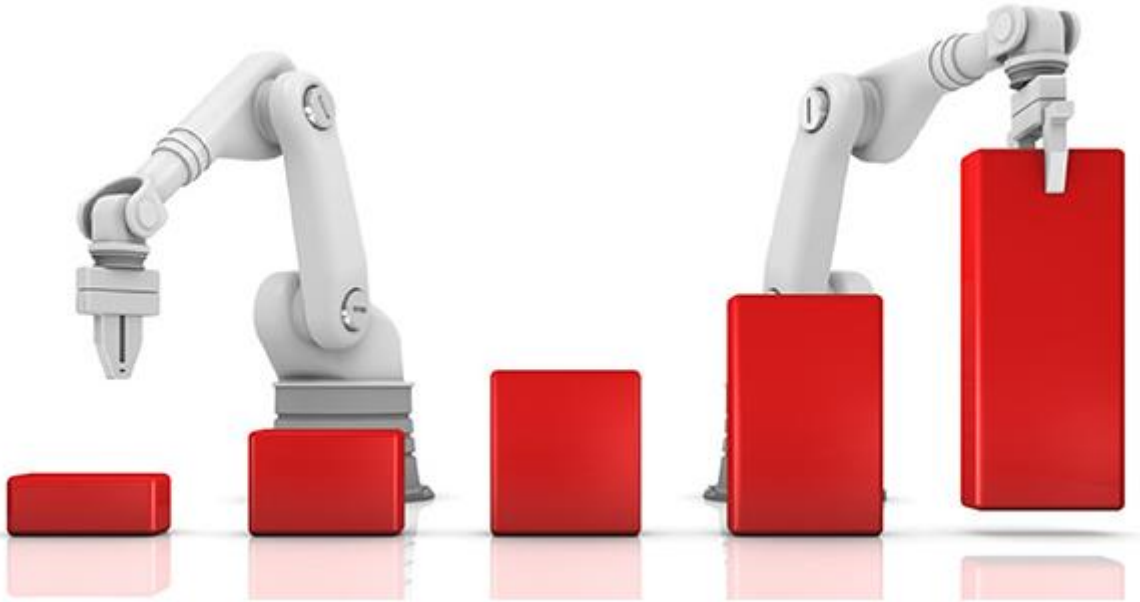


DevOps Automation on OCI Workshop

Agenda

- DevOps Introduction
- Challenges in DevOps
- Automation Tooling (Terraform, PSM)
- Workshop Overview
- Workshop Terraform Templates
- Labs

Goals and Objectives

- Understand DevOps automation concepts
- Describe the 4 categories of automation
- Compare and contrast different automation tools and categories
- Use Terraform and CLI to create, discover, and terminate resources
- Use Terraform with the OCI provider to provision IaaS resources, PaaS services
- Use Terraform with CLI to configure and deploy applications

DevOps Intro

Core Business Values of DevOps

FASTER RELEASES

- Quickly align with business requirements
- Increase accuracy of releases - avoid downtime

SAVE MONEY

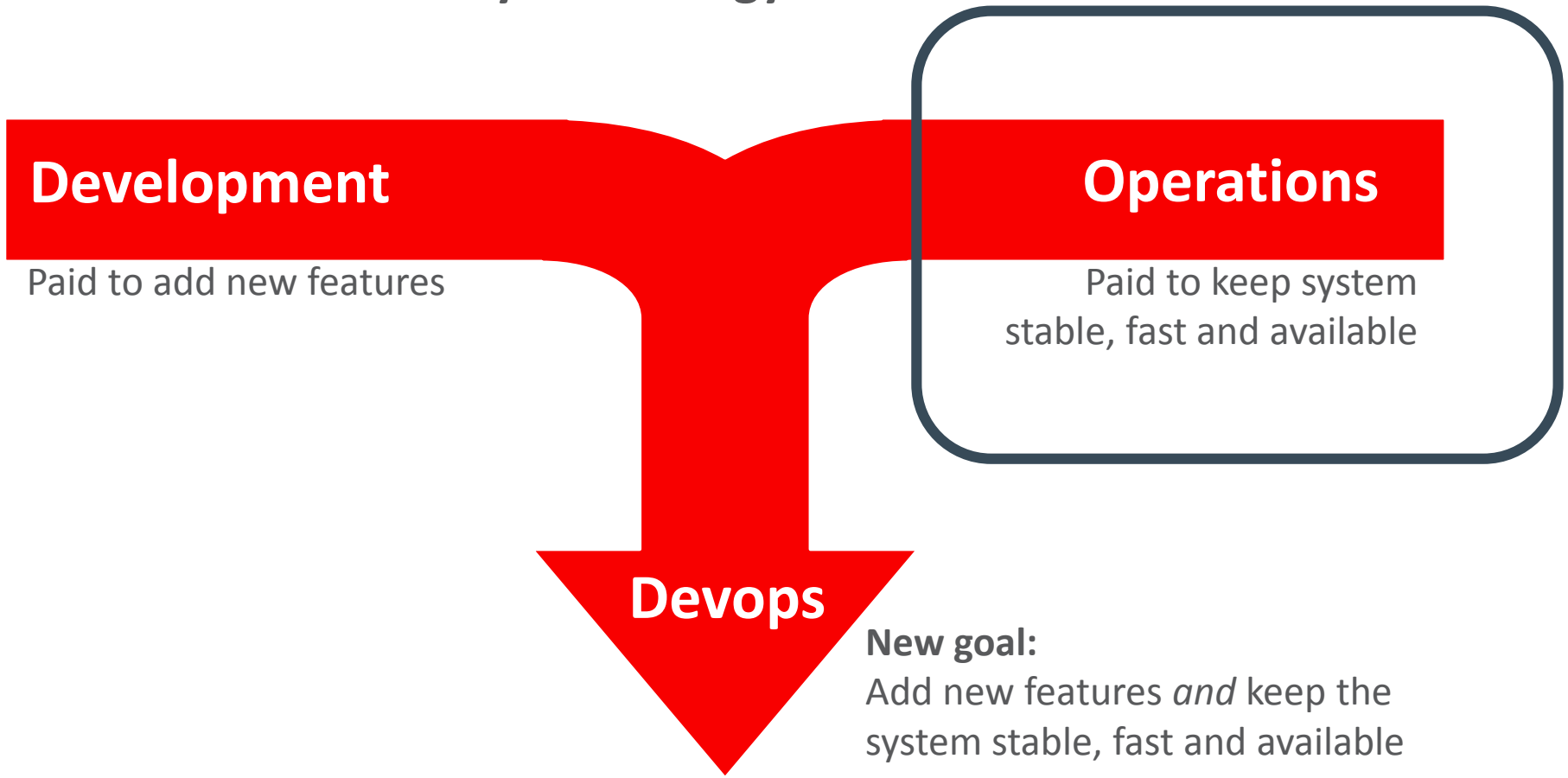
- Automate manual processes to reduce OPEX
- Prevent human error and reducing downtime

FOCUS ON BUSINESS

Allow high value employees to focus on higher value activities

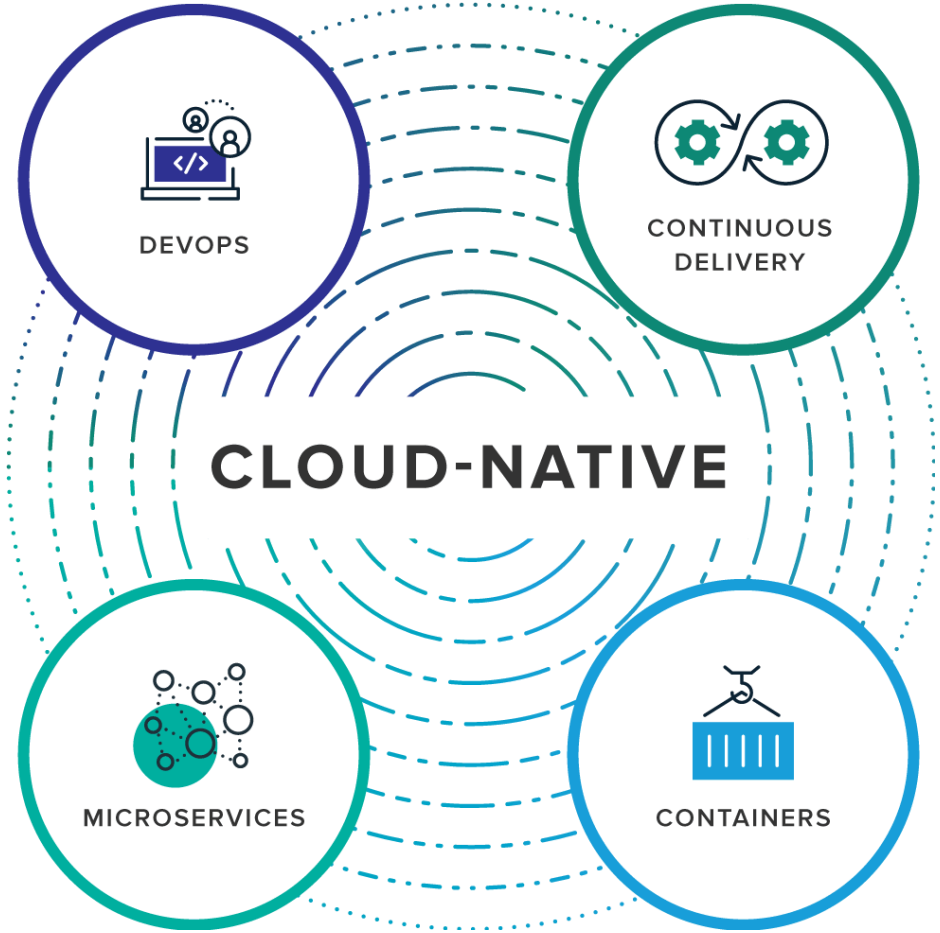
DevOps Principles

Cultural movement enabled by technology



New goal:
Add new features *and* keep the system stable, fast and available

DevOps

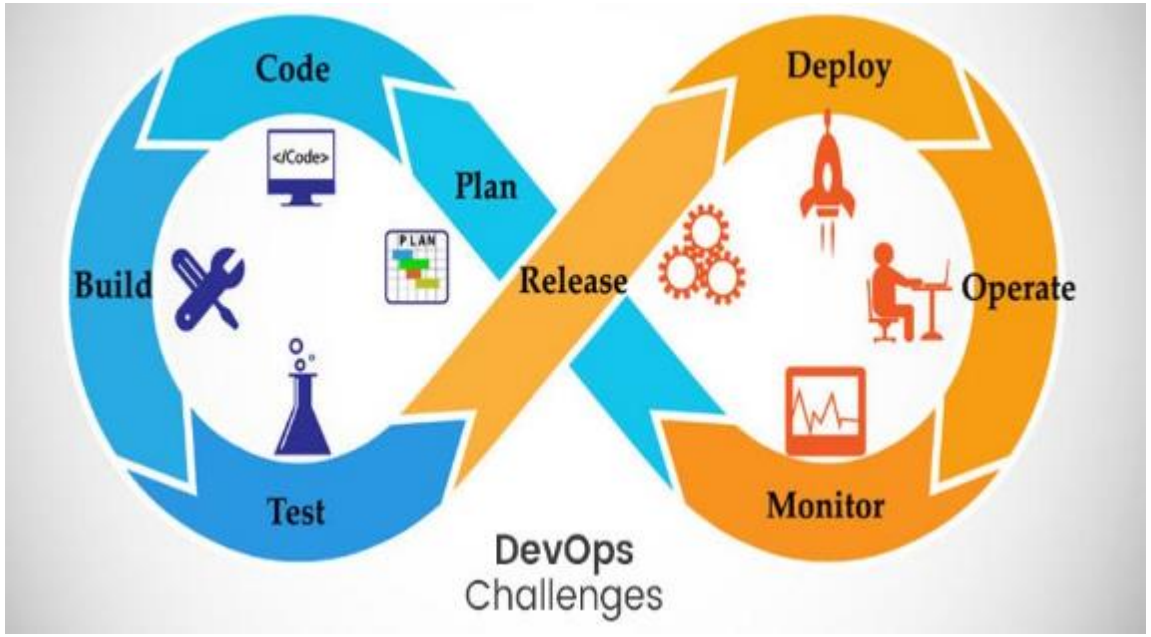


- Fully exploits the advantages of the cloud computing delivery model
- *Cloud Native* is not a specific workload, neither does it just apply to Development, but also a design & philosophy for Operations
- Oracle Cloud is providing the building blocks to enable higher value abstractions, automation and services
- This workshop provides the starting template for DevOps Automation

Challenges in DevOps

Top 10 DevOps Challenges

- Environment provisioning
- Manual testing
- No DevOps center of excellence
- Test data
- Manual deployments
- Planning in a DevOps environment
- DevOps and suppliers
- DevOps and governance
- No integrated tools architecture
- Manual releases



https://techbeacon.com/

The Power of Automation

4 Categories Automation Tools

1. Infrastructure Provisioning
   - Terraform, cloud formation, heat…
2. Server Templating
   - Packer, Vagrant, Docker…
3. Configuration Management
   - chef, puppet, ansible…
4. Ad hoc scripts
   - Shell scripts, CLI

Automation Tooling

Terraform – Built By HashiCorp

- Create and Manage infrastructure as Code
- Provisioning tool for managing Infrastructure Resources and Lifecycle
  - Provision
  - Update
  - Destroy
- Open Source Software with wide adoption in the market
  - Written in Go
- HCL - Hashi Configuration Language
  - simple markup format & JSON interoperable
- Enterprise support for Terraform available from HashiCorp

Terraform provider for Oracle Cloud Infrastructure

- https://github.com/oracle/terraform-provider-oci
- Maintained by OCI team and feature parity with OCI APIs
- Recommended way for deploying and managing stacks on OCI
- OCI Services supported
  - Core Services (Networking, Compute, Block Volume)
  - Database
  - DNS
  - File Storage
  - IAM
  - Load Balancing
  - Object Storage

OCI Terraform Samples

- https://github.com/oracle/terraform-provider-oci/tree/master/docs/examples
- https://github.com/oracle/terraform-provider-oci/tree/master/docs/solutions
- 1 click deploy to OCI
- Contains ~20 templates for basic resource management
- Additional templates for use-case specific stacks, e.g. MongoDB, Kubernetes

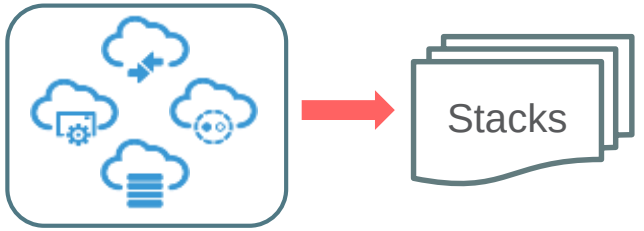
Terraform Provider for Oracle Cloud Platform

- Supported Oracle PaaS
  - Oracle Database Cloud Service
  - Oracle Java Cloud Service
  - MySQL (coming soon)
  - ACCS (coming soon)
- Supports creation and lifecycle management of Oracle PaaS
- Supports both OCI and OCI Classic

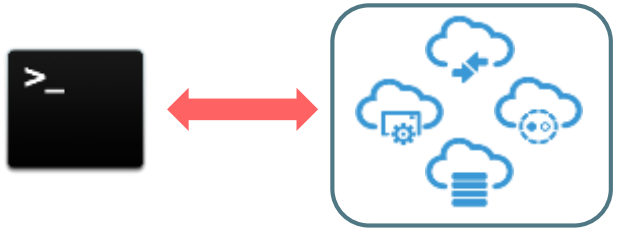
PaaS Provisioning

- Oracle PaaS Service Manager provides
  - Service Automation
  - Service and Stack Provisioning
  - API/CLI for DevOps

Cloud formations With Stack Manager



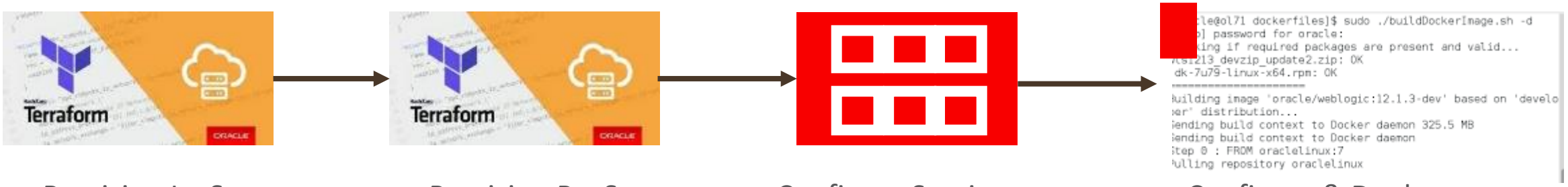
CLI for DevOps



Overview of the CLI

- The CLI is an essential tool for managing your OCI resources. It provides much of the same functionality found in the console, and extended functionality through the use of scripts.
- Built with the Python SDK
- Compatible with Python 2.7.5+ or 3.5+
- Compatible with Mac, Windows, and Linux
- Direct OCI API interaction

Putting It All Together with Automation in Oracle Cloud



Provision IaaS

Provision PaaS

Configure Services & Connectivity

Configure & Deploy Application

Putting It All Together with Automation in Oracle Cloud

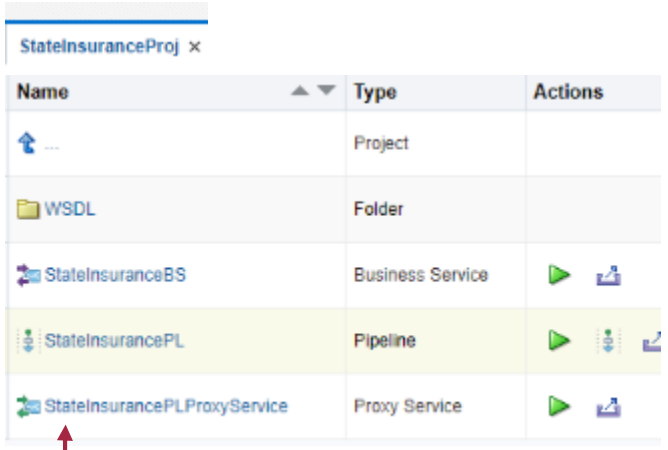
- Provision Oracle Cloud Service Instances
  - IaaS: Terraform
  - PaaS: Terraform
  - PaaS: PaaS Service Manager
- Configure Oracle Cloud Service Instances
  - Configure Service Instances
  - Configure Connectivity and Interconnectivity between instances
- Configure & Deploy Applications
  - Configure Application Deployment Parameters
  - Application startup
- Run Validation tests
- Return Environment Access Info

Workshop Overview

Workshops Overview

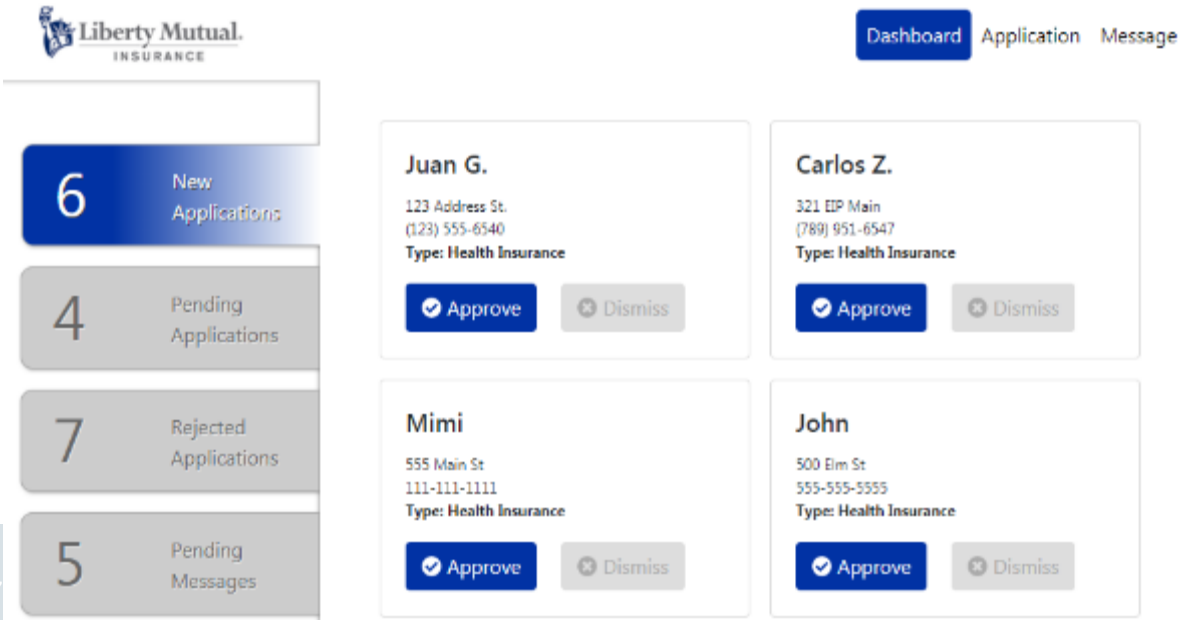
- Focus on **Provisioning of IaaS and PaaS** and **Applications Configuration/Deployment**
- Leverage Terraform & other configuration and deployment tools to provide ENVaaS to end users
- Targeting Real-World Applications that are complex and use many services like HIX
- Components used including vcn, compute, OCI database, JCS, SOACS and Docker Container running on compute.
- Tools: Terraform, PSM, Stack Manager, wlst, sqlplus and scripts

Workshop Sample Application – HHS Application

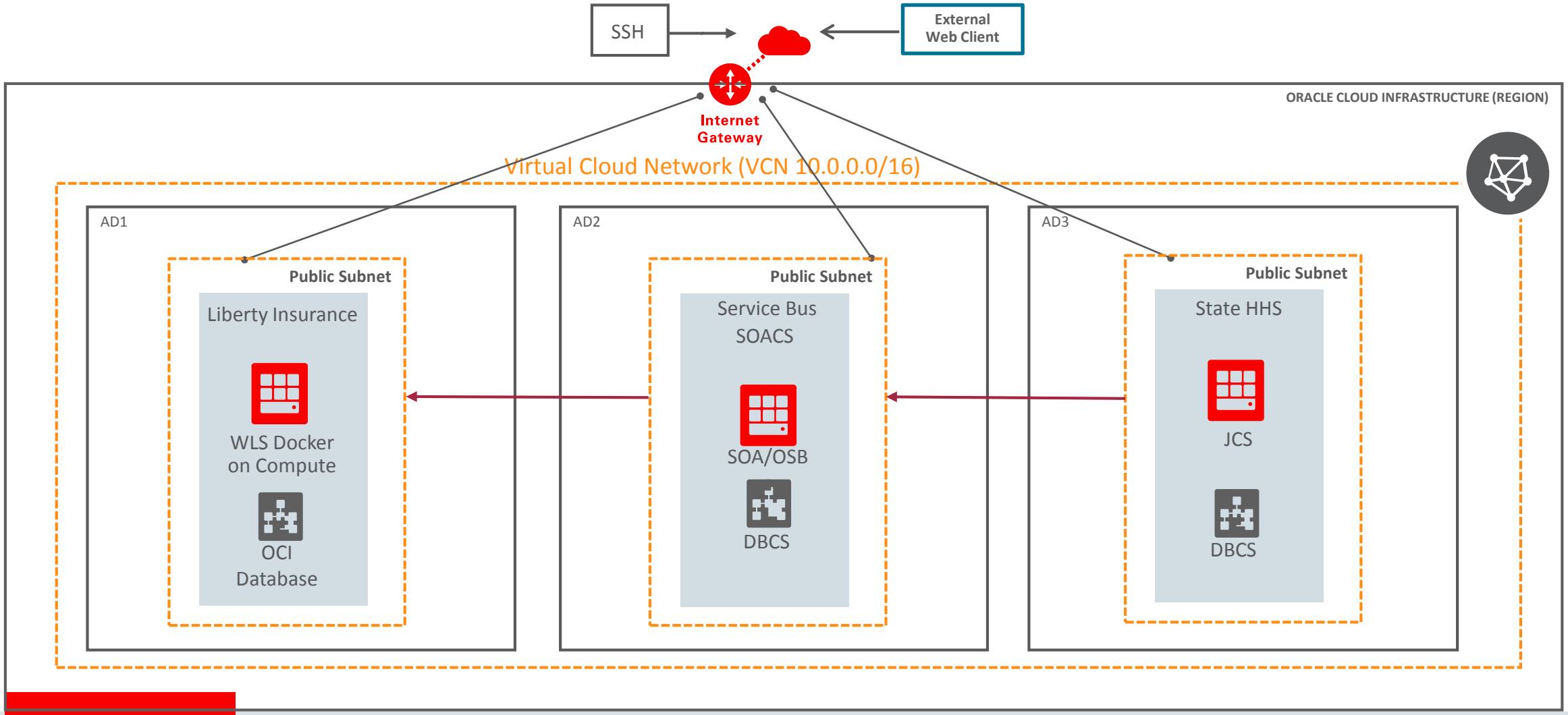


WS/SOAP Call to Liberty Insurance WS

WS/SOAP Call to
Service Bus
Proxy Service



Workshop Sample Application Architecture



Workshop Terraform Templates

Terraform Templates

- Lab 1 – Provisioning Environment
  - **env-var**
  - main.tf
    - module storage-swift
    - module vcn
    - module compute-instance
    - module database
    - module docker-config
    - module paas-config
    - output compute-instance public ip
    - output database public ip
  - provider.tf
  - vars.tf

env-var

```
### Authentication details
export TF_VAR_user="cloud.admin"
export TF_VAR_password="<password for cloud.admin>"
export TF_VAR_domain="<Identity Service Id: idcs-.........>"
export TF_VAR_tenancy="<Cloud Account Name: gse000#### >"
export TF_VAR_object_storage_user="gse-admin_ww@oracle.com"

### Public/private keys used on the instance
export TF_VAR_ssh_public_key_path=~/.ssh/id_rsa.pub
export TF_VAR_ssh_public_key=$(cat ~/.ssh/id_rsa.pub)
export TF_VAR_ssh_private_key=$(cat ~/.ssh/id_rsa)
export TF_VAR_ssh_authorized_private_key=$(cat ~/.ssh/id_rsa)

### Authentication details for oci provider
export TF_VAR_tenancy_ocid="<OCI tenancy ocid>"
export TF_VAR_user_ocid="<OCI user ocid for user: gse-admin_ww@oracle.com >"
export TF_VAR_fingerprint="27:3e:ea:41:6b:25:5d:23:52:ec:7b:ce:b6:98:19:f3"
export TF_VAR_private_key_path=~/.oci/oci_api_key.pem
export TF_VAR_region="<OCI Home Region>"
export TF_VAR_compartment_ocid="<OCI compartment ocid for Demo compartment>"
export TF_VAR_paas_compartment_ocid="<OCI compartment ocid for ManagedCompar
tmentForPaaS compartment>"
export TF_VAR_swift_password="<generated swift password for gse-admin_ww@ora
cle.com user>"
export TF_VAR_subscription_id="<Subscription ID>"
```

Terraform Templates

- Lab 1 – Provisioning Environment
  - env-var
  - **main.tf**
    - **module storage-swift**
    - **module vcn**
    - **module compute-instance**
    - module database
    - module docker-config
    - module paas-config
    - output compute-instance public ip
    - output database public ip
  - provider.tf
  - vars.tf

main.tf

```
module "object_storage" {
  source = "../modules/storage-swift"
  bucket_names = "${var.bucket_names}"
  env_prefix = "${var.env_prefix}"
  compartment_id = "${var.compartment_ocid}"
}

module "vcn" {
  source = "../modules/vcn"
  tenancy_ocid = "${var.tenancy_ocid}"
  compartment_ocid = "${var.compartment_ocid}"
  dns_vcn = "${var.env_prefix}${var.dns_vcn}"
  vcn_display = "${var.env_prefix}${var.vcn_display}"
}

module "compute" {
  source = "../modules/compute-instance"
  tenancy_ocid = "${var.tenancy_ocid}"
  compartment_ocid = "${var.compartment_ocid}"
  ssh_public_key = "${var.ssh_public_key}"
  ssh_private_key = "${var.ssh_authorized_private_key}"
  instance_shape = "${var.instance_shape}"
  subnet = "${module.vcn.subnet1_ocid}"
  name = "${var.env_prefix}${var.compute_name}"
  availability_domain = "${module.vcn.subnet1_ad}"
}
```

Terraform Templates

- Lab 1 – Provisioning Environment
  - env-var
  - **main.tf**
    - module storage-swift
    - module vcn
    - module compute-instance
    - **module database**
    - **module docker-config**
    - module paas-config
    - output compute-instance public ip
    - output database public ip
  - provider.tf
  - vars.tf

main.tf

```
module "database" {
  source = "../modules/database"
  tenancy_ocid = "${var.tenancy_ocid}"
  compartment_ocid = "${var.compartment_ocid}"
  availability_domain = "${module.vcn.subnet1_ad}"
  SubnetOCID = "${module.vcn.subnet1_ocid}"
  ssh_public_key = "${var.ssh_public_key}"
  DBNodeDomainName = "${module.vcn.subnet1_label}.${var.env
_prefix}${var.dns_vcn}.oraclevcn.com"
  DBNodeShape = "${var.DBNodeShape}"
  DBAdminPassword = "${var.DBAdminPassword}"
  DBName = "${var.DBName}"
  DBNodeDisplayName = "${var.env_prefix}${var.DBName}"
  PDBName = "${var.PDBName}"
  ssh_private_key = "${var.ssh_authorized_private_key}"
}

module "docker-config" {
  source = "../modules/docker-config"
  tenancy_ocid = "${var.tenancy_ocid}"
  compartment_ocid = "${var.compartment_ocid}"
  public-ip = "${module.compute.public-ip}"
  ssh_private_key = "${var.ssh_authorized_private_key}"
  config_src_dir = "${var.config_src_dir}"
}
```

ORACLE

Terraform Templates

- Lab 1 – Provisioning Environment
  - env-var
  - **main.tf**
    - module storage-swift
    - module vcn
    - module compute-instance
    - module database
    - module docker-config
    - **module paas-config**
    - **output compute-instance public ip**
    - **output database public ip**
  - provider.tf
  - vars.tf

main.tf

```
module "paas" {
  source = "../modules/paas-config"
  user = "${var.user}"
  db_password = "${var.DBAdminPassword}"
  password = "${var.password}"
  domain = "${var.domain}"
  jcs_subnet = "${module.vcn.subnet1_ocid}"
  soacs_subnet = "${module.vcn.subnet2_ocid}"
  region = "${var.region}"
  tenancy_ocid = "${var.tenancy_ocid}"
  ssh_public_key_path = "${var.ssh_public_key_path}"
  object_storage_user = "${var.object_storage_user}"
  swift_password = "${var.swift_password}"
  OTDShape = "${var.OTDShape}"
  SOAShape = "${var.SOAShape}"
  SOADBShape = "${var.SOADBShape}"
  JCSShape = "${var.JCSShape}"
  DBShape = "${var.DBShape}"
  tenancy = "${var.tenancy}"
  buckets = "${module.object_storage.names}"
  jcs_ad = "${module.vcn.subnet1_ad}"
  soacs_ad = "${module.vcn.subnet2_ad}"
  env_prefix = "${var.env_prefix}"
}
output "Compute Public IP" {
  value = "${module.compute.public-ip}"
}

output "DB Public IP" {
  value = "${module.database.DBNodePublicIP}"
}
```

Terraform Templates

- Lab 1 – Provisioning Environment
  - env-var
  - main.tf
    - module storage-swift
    - module vcn
    - module compute-instance
    - module database
    - module docker-config
    - module paas-config
    - output compute-instance public ip
    - output database public ip
  - **provider.tf**
  - vars.tf

provider.tf

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

Terraform Templates

- Lab 1 – Provisioning Environment
  - env-var
  - main.tf
    - module storage-swift
    - module vcn
    - module compute-instance
    - module database
    - module docker-config
    - module paas-config
    - output compute-instance public ip
    - output database public ip
  - provider.tf
  - **vars.tf**

vars.tf

```
variable "tenancy_ocid" {}
variable "user_ocid" {}
variable "fingerprint" {}
variable "private_key_path" {}
variable "region" {}
variable "ssh_public_key" {}
variable "ssh_public_key_path" {}
variable "ssh_authorized_private_key" {}
variable "compartment_ocid" {}
variable "paas_compartment_ocid" {}

variable "subscription_id" {}
variable "user" {}
variable "password" {}
variable "domain" {}
variable "tenancy" {}
variable "object_storage_user" {}

variable "swift_password" {}
```

Terraform Templates

- Lab 1 – Provisioning Environment
  - env-var
  - main.tf
    - module storage-swift
    - module vcn
    - module compute-instance
    - module database
    - module docker-config
    - module paas-config
    - output compute-instance public ip
    - output database public ip
  - provider.tf
  - **vars.tf**

vars.tf

```
variable "env_prefix" {
  default = "lab1"
}

variable "bucket_names" {
  default = ["jcs_backup", "jcs_dbcs_backup", "soacs_backup"
, "soacs_dbcs_backup" ]
}


variable "dns_vcn" {
  default="tfvcn"
}
variable "vcn_display" {
  default="DevOpsVCN"
}


variable "compute_name" {
  default="DevOps-Instance"
}
variable "instance_shape" {
  default="VM.Standard2.1"
}
```

Terraform Templates

- Lab 1 – Provisioning Environment
  - env-var
  - main.tf
    - module storage-swift
    - module vcn
    - module compute-instance
    - module database
    - module docker-config
    - module paas-config
    - output compute-instance public ip
    - Output database public ip
  - provider.tf
  - **vars.tf**

vars.tf

```
variable "DBNodeShape" {
    default = "VM.Standard1.2"
}
variable "DBAdminPassword" {
    default = "STateinsurance12#_"
}

# OracleDB SID
variable "DBName" {
    default = "TFdb"
}

variable "PDBName" {
  default = "pdbName"
}
```

Terraform Templates

- Lab 1 – Provisioning Environment
  - env-var
  - main.tf
    - module storage-swift
    - module vcn
    - module compute-instance
    - module database
    - module docker-config
    - module paas-config
    - output compute-instance public ip
    - output database public ip
  - provider.tf
  - **vars.tf**

vars.tf

```
variable "config_src_dir" {
  default="/app"
}

variable "DBShape" {
  default="VM.Standard1.2"
}
variable "JCSShape" {
  default="VM.Standard2.1"
}
variable "SOAShape" {
  default="VM.Standard1.2"
}
variable "SOADBShape" {
  default="VM.Standard2.1"
}
variable "OTDShape" {
  default="VM.Standard1.1"
}
```

Terraform Modules

- storage-swift
  - main.tf
    - create buckets from input variable bucket_names (list)
    - on destroy, run a script to bulk delete objects in bucket before destroy of the bucket
  - scripts/delete_objs_by_bucket.sh
  - output.tf
    - output bucket names
  - vars.tf
    - Input variables for the module
    - Any input variable without a default value is required when calling the module

```
data "oci_objectstorage_namespace" "t" {
}

resource "oci_objectstorage_bucket" "bucket" {
  compartment_id = "${var.compartment_id}"
  name = "${var.env_prefix}${var.bucket_names[count.index]}"
  namespace = "${data.oci_objectstorage_namespace.t.namespace}"
  count = "${length(var.bucket_names)}"

  provisioner "local-exec" {
    when = "destroy"
    command = "${path.module}/scripts/delete_objs_by_bucket.sh
${var.env_prefix}${var.bucket_names[count.index]}"
  }

}
output "names" {
  value = "${oci_objectstorage_bucket.bucket.*.name}"
}
```

Terraform Modules

- vcn
  - datasources.tf
    - Lookup AD
  - main.tf
    - virtual cloud network
    - internet gateway
    - route table
    - security list
    - public subnet in each AD
  - outputs.tf
    - output subnet id
    - output subnet dns label
    - output subnet AD
  - vars.tf

```
resource "oci_core_virtual_network" "TF_VCN" {
  cidr_block = "10.0.0.0/16"
  ...
}

resource "oci_core_internet_gateway" "TF_IG" {
  ...
}

resource "oci_core_route_table" "TF_RT" {
  ...
}

resource "oci_core_security_list" "TF_SL_Public" {
  ...
}

resource "oci_core_subnet" "TF_Public_SubnetAD1" {
  cidr_block = "10.0.1.0/24"
  ...
}

resource "oci_core_subnet" "TF_Public_SubnetAD2" {
  cidr_block = "10.0.2.0/24"
  ...
}

resource "oci_core_subnet" "TF_Public_SubnetAD3" {
  cidr_block = "10.0.3.0/24"
  ...
}
output "subnet1_ocid" {
  value = "${oci_core_subnet.TF_Public_SubnetAD1.id}"
}

output "subnet2_ocid" {
  value = "${oci_core_subnet.TF_Public_SubnetAD2.id}"
}

output "subnet3_ocid" {
  value = "${oci_core_subnet.TF_Public_SubnetAD3.id}"
}
```

Terraform Modules

- compute-instance
  - datasources.tf
    - Lookup AD
  - compute.tf
    - Create compute instance
  - output.tf
    - output public ip
  - vars.tf

```
resource "oci_core_instance" "devops" {
  availability_domain = "${var.availability_domain}"
  compartment_id      = "${var.compartment_ocid}"
  #image               = "${var.image_ocid}"
  shape               = "${var.instance_shape}"
  display_name = "${var.name}"

  create_vnic_details {
    subnet_id = "${var.subnet}"
    hostname_label = "${var.name}"
  }

  source_details {
    source_type = "image"
    source_id = "${var.image_ocid}"
  }

  metadata = {
    "ssh_authorized_keys" = "${var.ssh_public_key}"
  }

  timeouts = {
    "create" = "60m"
  }
}

output "public-ip" {
  value = "${oci_core_instance.devops.public_ip}"
}
```

Terraform Modules

- docker-config
  - main.tf
    - Copy scripts, docker files and WebLogic installation packages
    - Configure docker container
    - Install WebLogic
    - Configure WebLogic Domain
    - Start WebLogic Server
  - scripts/install_weblogic.sh
  - vars.tf

```
resource "null_resource" "config-scripts" {

  provisioner "file" {
    connection {
      host = "${var.public-ip}"
      user = "ubuntu"
      private_key = "${var.ssh_private_key}"
    }
    source      = "${path.module}/scripts/"
    destination = "/tmp/"
  }
}

resource "null_resource" "config-installer" {

  provisioner "file" {
   connection {
     host = "${var.public-ip}"
     user = "ubuntu"
     private_key = "${var.ssh_private_key}"
   }
   source      = "${var.config_src_dir}/installer/"
   destination = "/tmp/"
  }
}

resource "null_resource" "weblogic-config" {

  depends_on = ["null_resource.config-installer", "null_resource.config-scripts"]

  provisioner "remote-exec" {
    connection {
      host= "${var.public-ip}"
      user = "ubuntu"
      private_key = "${var.ssh_private_key}"
    }

    inline = [
      "chmod +x /tmp/install_weblogic.sh",
      "sudo /tmp/install_weblogic.sh"
    ]
  }
}
```

Terraform Modules

- Database
  - datasources.tf
    - Lookup AD
    - Get Vnic of DB Node
  - main.tf
    - Create database system
  - config.tf
    - Run scripts to configure pdb by creating schema and tables needed
  - Scripts
    - db_config.sh
    - StateInsurance.sql
  - output.tf
    - output public ip
  - vars.tf

```
resource "oci_database_db_system" "TFDBNode" {
  availability_domain = "${var.availability_domain}"
  compartment_id = "${var.compartment_ocid}"
  cpu_core_count = "${substr(var.DBNodeShape, 13, -1)}"
  database_edition = "${var.DBEdition}"
  db_home {
    database {
      "admin_password" = "${var.DBAdminPassword}"
      "db_name" = "${var.DBName}"
      "pdb_name" = "${var.PDBName}"
    }
    db_version = "${var.DBVersion}"
  }
  shape = "${var.DBNodeShape}"
  subnet_id = "${var.SubnetOCID}"
  ssh_public_keys = ["${var.ssh_public_key}"]
  hostname = "${var.DBName}"
  data_storage_size_in_gb = "${var.DataStorageSizeInGB}"
  node_count = "${var.NodeCount}"
  display_name = "${var.DBNodeDisplayName}"
}
```

```
resource "null_resource" "db-config" {
  provisioner "file" {
    connection {
      host= "${data.oci_core_vnic.DBNodeVnic.public_ip_address}"
      user = "opc"
      private_key = "${var.ssh_private_key}"
    }
    source = "${path.module}/scripts/"
    destination = "/tmp"
  }

  provisioner "remote-exec" {
    connection {
      host= "${data.oci_core_vnic.DBNodeVnic.public_ip_address}"
      user = "opc"
      private_key = "${var.ssh_private_key}"
    }

    inline = [
      "chmod 777 /tmp/db_config.sh",
      "chmod 666 /tmp/StateInsurance.sql",
      "sudo su - oracle -c \"/tmp/db_config.sh ${var.DBName} ${v
ar.DBNodeDomainName} ${var.PDBName} \""
    ]
  }
}
```

```
output "DBNodePublicIP" {
  value = ["${data.oci_core_vnic.DBNodeVnic.public_ip_address}"]
}
```

ORACLE

Terraform Templates

- Lab 2 – Configure/Deploy Applications
  - app_config.tf.template
    - Add the following section

```
module "app-config" {
  source = "../modules/app-config"
  wlst = "/app/fmw/oracle_common/common/bin/wlst.sh"
  liberty_ip = "${module.compute.public-ip}"
  osb_ip = "${trimspace(module.get-paas-info.soa_public_ip)}"
  jcs_ip = "${trimspace(module.get-paas-info.jcs_public_ip)}"
  password = "${var.DBAdminPassword}"
  dbconn = "jdbc:oracle:thin:@//${module.database.DBNodePublicIP[0]}
:1521/${var.PDBName}.${module.vcn.subnet2_label}.${var.env_prefix}${
var.dns_vcn}.${var.oraclevcn}"
  targets = "${local.jcs_cluster}"
  ssh_private_key = "${var.ssh_authorized_private_key}"
}
```

  - app_config.tf.solution

```
module "get-paas-info" {
  source = "../modules/get-paas-info"
  paas_compartment_id = "${var.paas_compartment_ocid}"
  jcs_display_name = "${var.subscription_id}|JaaS|${var.env_prefix}JCSDBCSStackJ
CS|wls|vm-1"
  soa_display_name = "${var.subscription_id}|SOA|${var.env_prefix}SOAStackSOACS|
wls|vm-1"
}

locals {
  jcsname =  "${var.env_prefix}JCSDBCSStackJCS"
  jcs_cluster = "${substr(local.jcsname, 0, 8)}_cluster"
}


#####################################################################
#
# call module app-config here
#
#####################################################################



output "LibertyInsurance App Url" {
  value = "http://${module.compute.public-ip}:7001/LibertyInsurance-WebServiceAp
p-context-root/"
}

output "StateGov App Url" {
  value = "http://${trimspace(module.get-paas-info.jcs_public_ip)}/StateGov-WebS
ervice-context-root/"
}
```

Terraform Modules

- app-config
  - main.tf
    - **On WebLogic Server running on Docker Container, configure JDBCS Data Source to OCI Database and deploy Liberty Insurance App**
    - **On SOACS, Import Service Bus Project**
    - On JCS, deploy State HHS app
  - config_deploy_liberty_app.py
  - LibertyInsurance-WebServiceApp-context-root.war
  - import_sbconfig.py
  - sbconfig.jar

```
resource "null_resource" "liberty-app-config" {

  provisioner "local-exec" {
    command = "${var.wlst} ${path.module}/config_deploy_liberty_app.py t3://${va
r.liberty_ip}:7001 welcome1 ${var.password} ${path.module}/${var.liberty_warfile
} ${var.dbconn}  "
  }
}

resource "null_resource" "osb-proxy-config" {
  depends_on = ["null_resource.liberty-app-config"]

  provisioner "remote-exec" {
    connection {
      host= "${var.osb_ip}"
      user = "opc"
      private_key = "${var.ssh_private_key}"
    }

    inline = [
      "sudo su -c \"echo ${var.liberty_ip}  LibertyWLS >> /etc/hosts \" "
    ]
  }
  provisioner "local-exec" {
    /*********************************************************************
      Note: this wlst must include required osb jar files in the classpath
            such as the following:
            OSB_HOME="/u01/fmw/osb"
            CLASSPATH=${OSB_HOME}/lib/modules/oracle.servicebus.configfwk.jar:${
OSB_HOME}/lib/modules/oracle.servicebus.kernel-api.jar:${OSB_HOME}/lib/modules/o
racle.servicebus.configfwk-wls.jar:${OSB_HOME}/lib/modules/oracle.servicebus.ker
nel-wls.jar:${CLASSPATH}

    *********************************************************************/

    command = "${var.wlst} ${path.module}/import_sbconfig.py t3://${var.osb_ip}:
9001 weblogic ${var.password} ${path.module}/${var.sbconfig_jarfile} "
  }
}
```

Terraform Modules

- app-config
  - deploy_state_app.py
  - StateGov-WebService-context-root.war
  - vars.tf

```
resource "null_resource" "state-app-config" {
  depends_on = ["null_resource.osb-proxy-config"]

  provisioner "remote-exec" {
    connection {
      host= "${var.jcs_ip}"
      user = "opc"
      private_key = "${var.ssh_private_key}"
    }

    inline = [
      "sudo su -c \"echo ${var.osb_ip}  soastacksoacs   >> /etc/hosts \""
    ]
  }

  provisioner "local-exec" {
    command = "${var.wlst} ${path.module}/deploy_state_app.py t3://${var.jcs_ip}
:9001 ${var.password} ${var.targets} ${path.module}/${var.state_warfile} "
  }
}
```

Labs

Lab Environment

- Lab Environment Access Details

Workshop VM Access	
Public IP	129.213.85.88
Username	devop00
OCI Cloud Account Access	
Identity Domain	gse00014442
Login Username	cloud.admin
Login Password	sTeady@3Finger
OCI Console URL	https://console.us-ashburn-1.oraclecloud.com/#/a/
Cloud Service Dashboard URL	https://myservices-gse00014442.console.oraclecloud.com/mycloud/cloudportal/dashboard
Swift Password	Y}uneQp8GF9TNJsCwW){

- ssh keys
  - testdrive-private.ppk for putty
  - testdrive_unix.prv for ssh

Demo and Hands-on Lab 1

Lab1

45

Minutes

Demo and Hands-on Lab 2

Lab2

45

Minutes

Recap & Final Terraform for Oracle PaaS Demo

Thank you!

Q&A