Microsoft

SAP on Azure DevOps for SAP Workloads

Wednesday, October 21st, 2020

10am – 12 pm SGT

Nicolas Yuen & Ravi Gangampali
Microsoft APAC

IMPT NOTICE:

- If you choose to participate in this session using Microsoft Teams, your name, email address, phone number, and/or title may be viewable by other session participants.
- Please note that the training will not and cannot be recorded in alignment with Microsoft's policies

SAP on Azure Partner Enablement

Module Two – Week Two

Day 3 – SAP and DevOps

Nicolas Yuen
Cloud Solution Architect

Ravi Gangampalli
Cloud Solution Architect– SAP on Azure

Anjan Banerjee
Sr Customer Engineer– SAP on Azure
Azure Engineering

Check-in

We are happy to host you ☺

https://aka.ms/apac-enablement-check-in

Agenda

DevOps – quick recap

SAP deployment on Azure

SAP and Terraform

Infra As Code core concepts

Automation with Azure DevOps

DevOps quick Recap

What is DevOps

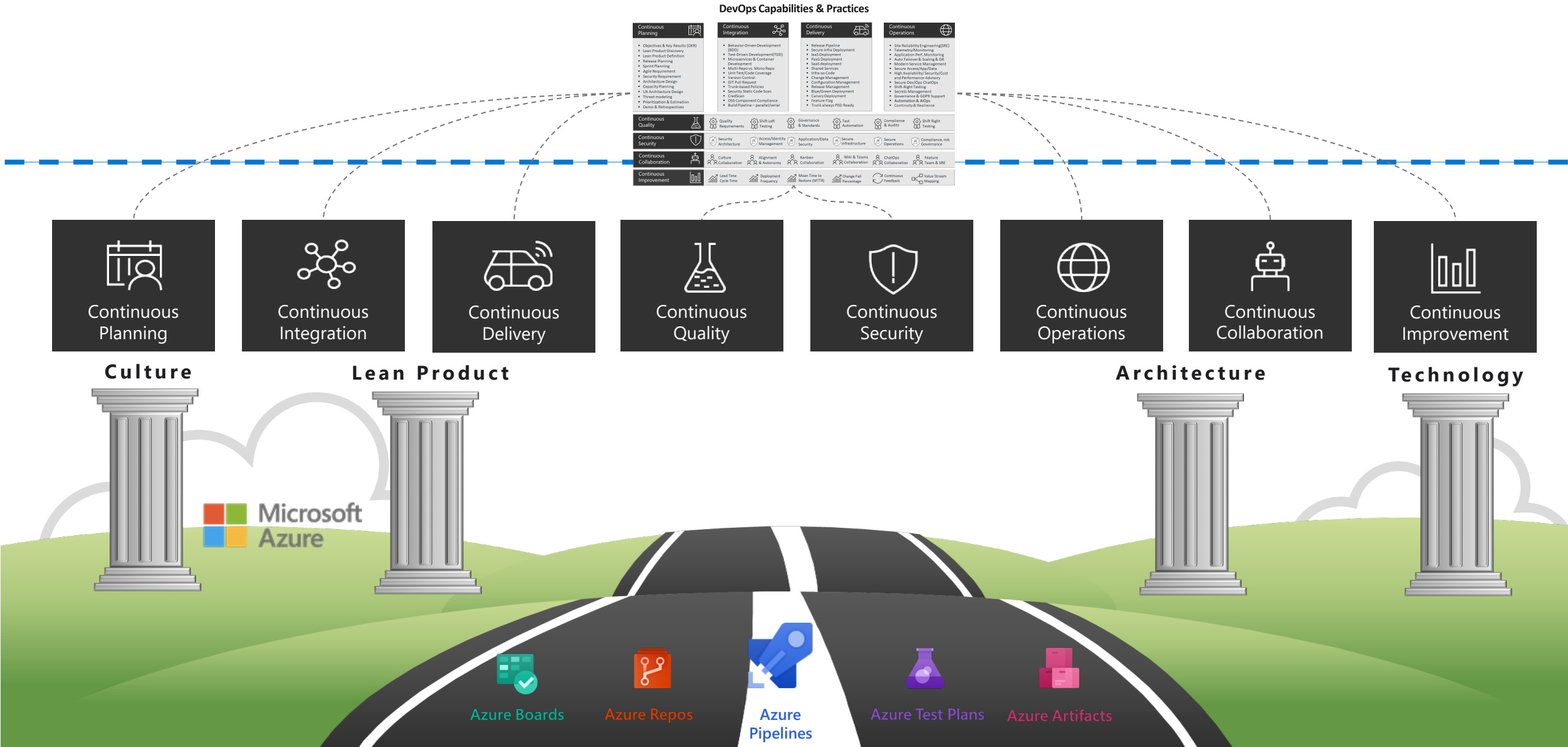
The contraction of “Dev” and “Ops” refers to replacing siloed Development and Operations to create multidisciplinary teams that work together with shared and efficient practices and tools.

Conway's law

- "Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure" Melvin Conway – 1968

- "T*he org chart of the company has a direct impact on the code*" Clément Rochas

DevOps Taxonomy

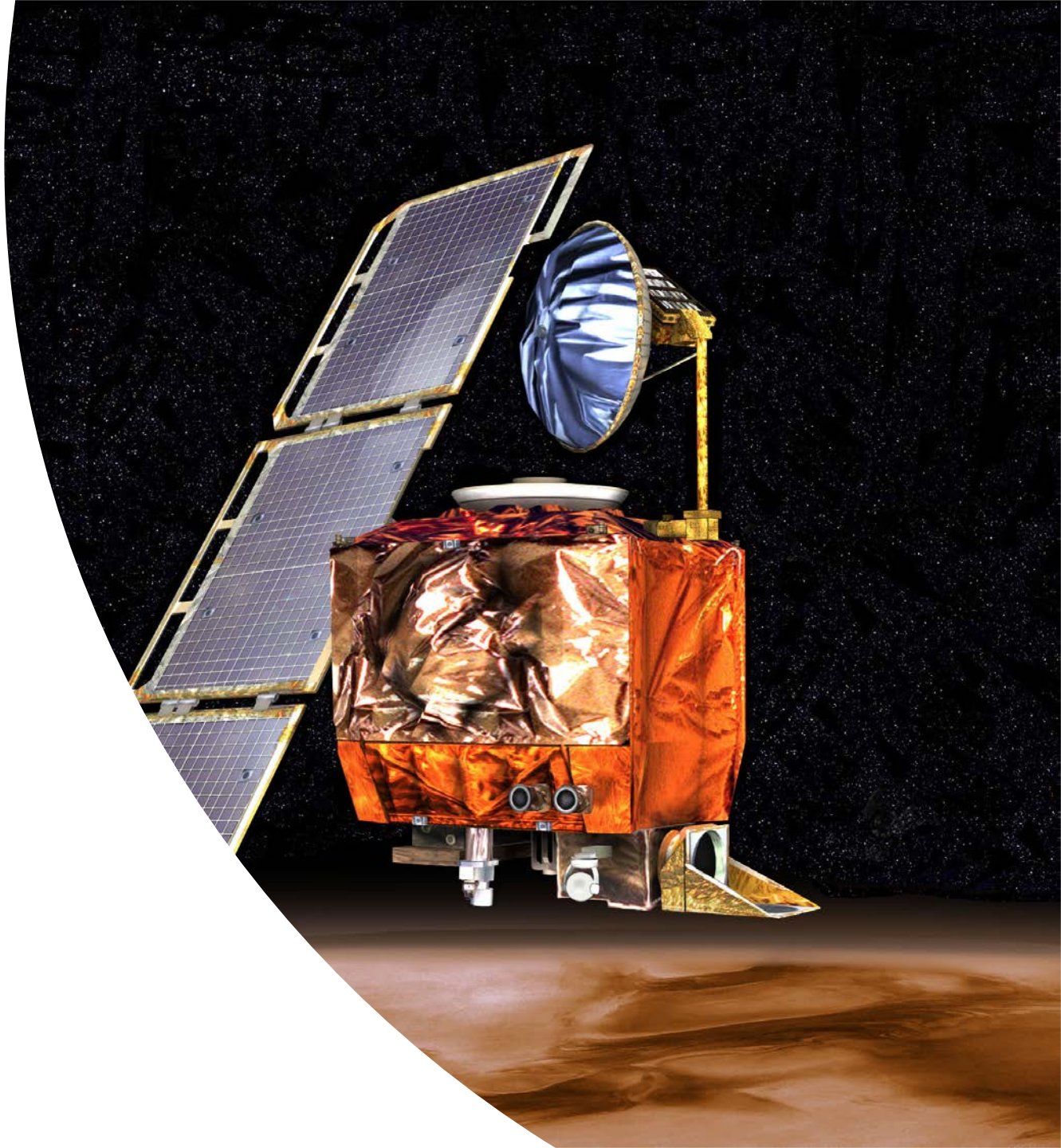


Mars Climate Orbiter

- **Mission**

  The craft was intended to be the first ever weather satellite for Mars, which would monitor the planet's atmosphere with its high-resolution camera.

- **What happened**
  - The craft burned up in the Martian atmosphere
  - $327.6 million loss for NASA
- **What went wrong**
  - Bug in the ground control software supplied by a third party
  - Calculations were performed manually rather than using the supplied software, due to file format errors and miscellaneous bugs



What are the Goals of Continuous Integration

The main purpose of Continuous Integration is to prevent developers stepping over each other code and eliminate integration issues.

CI Goals:

#1: Harness collaboration

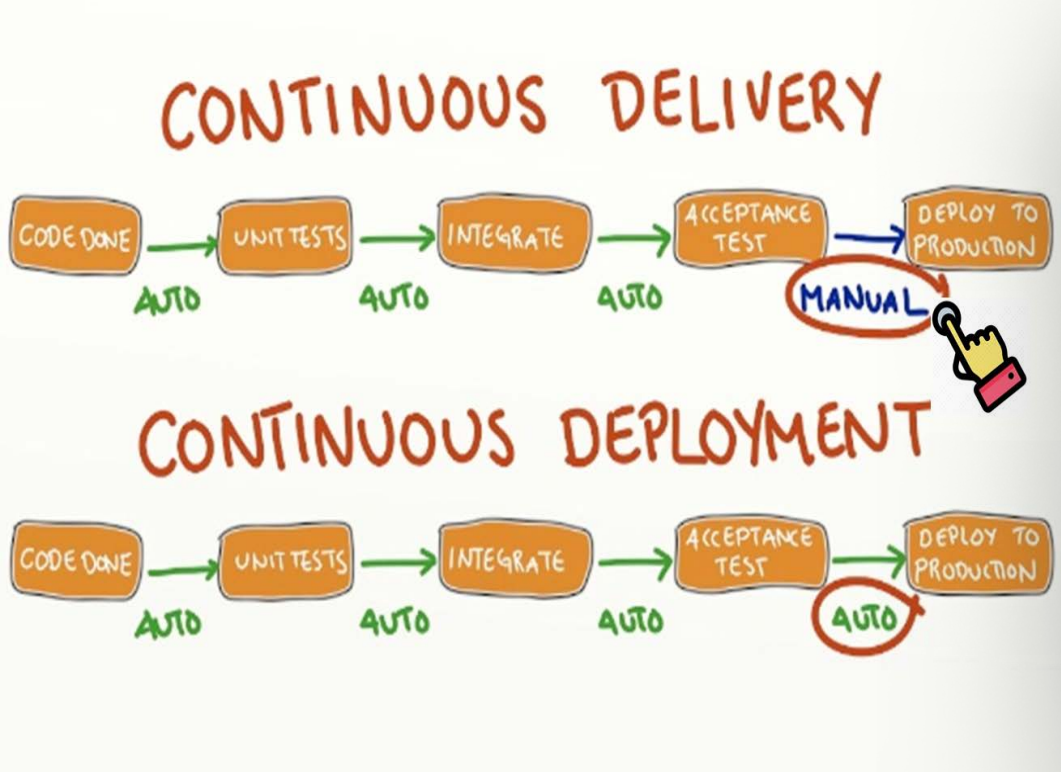
#2: Enable parallel development

#3: Minimize integration debt

#4: Act as a quality gate

#5: Automate Everything!

What is Continuous Delivery

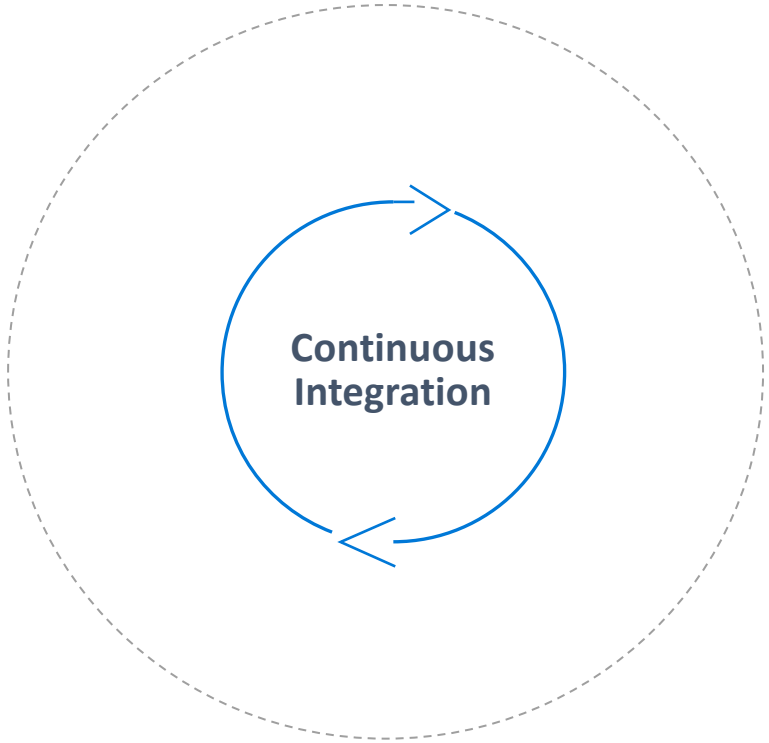


Continuous delivery is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time and, when releasing the software, doing so manually. It aims at building, testing, and releasing software with greater speed and frequency. The approach helps reduce the cost, time, and risk of delivering changes by allowing for more incremental updates to applications in production. A straightforward and repeatable deployment process is important for continuous delivery.

Source: https://en.wikipedia.org/wiki/Continuous_delivery

Continuous Integration Practices

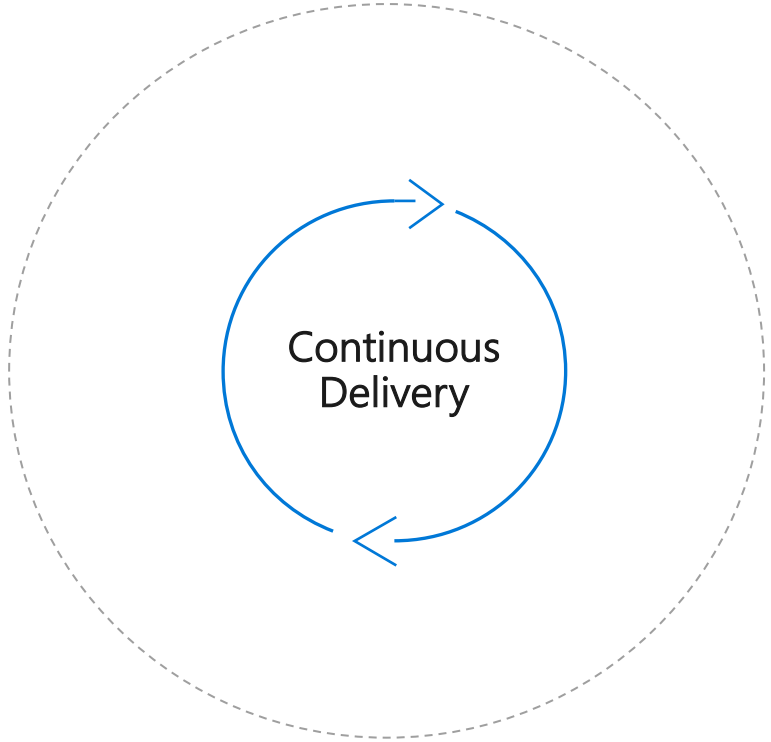
- Multi-Repo
- Trunk Based Branching
- Branch Policies
- Parallel Builds



- Culture & Collaboration
- Unit Testing
- Pull Requests
- Security Scanning

Continuous Delivery Practices

- Automated Builds
- Build Once
- Trunk Always Prod Ready
- Fix Immediately



- Everything in Source Control
- Feature Flags
- Blue/Green Deploy
- Canary Deployment

SAP deployment options

Methods for deploying SAP (and everything else) in Azure

Three over-arching methods.

- Azure Portal (manual)
- Scripts / SDKs (automation)
- Template based deployments (automation)

Methods for Azure resource deployments

Azure Portal (manual)

Pros:

- Browser based, quick setup, no fuss
- Nice for exploration and visual inspection & monitoring
- Fully featured

Cons:

- Everything is performed manually
- Error prone
- Lack of process integration (DevOps, ITSM)

Methods for Azure resource deployments

Scripts / SDKs (automation)

Pros:

- Process integration (DevOps / ITSM) possible
- Removes human / less error prone
- Unopinionated / total flexibility

Cons:

- Requires scripting knowledge / environment
- Complex logic needs to be hand built
- DIY error handling

Creating a RG, vnet, & VM with Scripts

- Powershell Example here: https://docs.microsoft.com/en-us/azure/virtual-machines/scripts/virtual-machines-windows-powershell-sample-create-vm
- CLI example here: https://docs.microsoft.com/en-us/azure/virtual-machines/scripts/virtual-machines-windows-cli-sample-create-vm

Methods for Azure resource deployments

Template based deployments (automation)

Pros:

- Process integration (DevOps / ITSM) possible
- Removes human / less error prone
- Handles some complex logic
- Options for state management

Cons:

- Requires templating knowledge / environment
- Opinionated and lack of full flexibility
- Some error handling

Template based deployments

Digging deeper on template based deployments.

- Azure Resource Manager templates or Terraform
- Declaration of desired infrastructure
- JSON or JSON like syntax
- Deploy, update, delete

Terraform

What is Terraform?

- Open source project
- Cross computing environment templating language
- Provision, Update, and Delete resources
- Authored in HashiCorp Configuration Language (HCL) or JSON

Terraform Example

```
resource "azurerm_resource_group" "testrg" {
    name = "resourceGroupName"
    location = "westus"
}

resource "azurerm_storage_account" "testsa" {
    name = "storageaccountname"
    resource_group_name = "testrg"
    location = "westus"
    account_tier = "Standard"
    account_replication_type = "GRS"
}
```

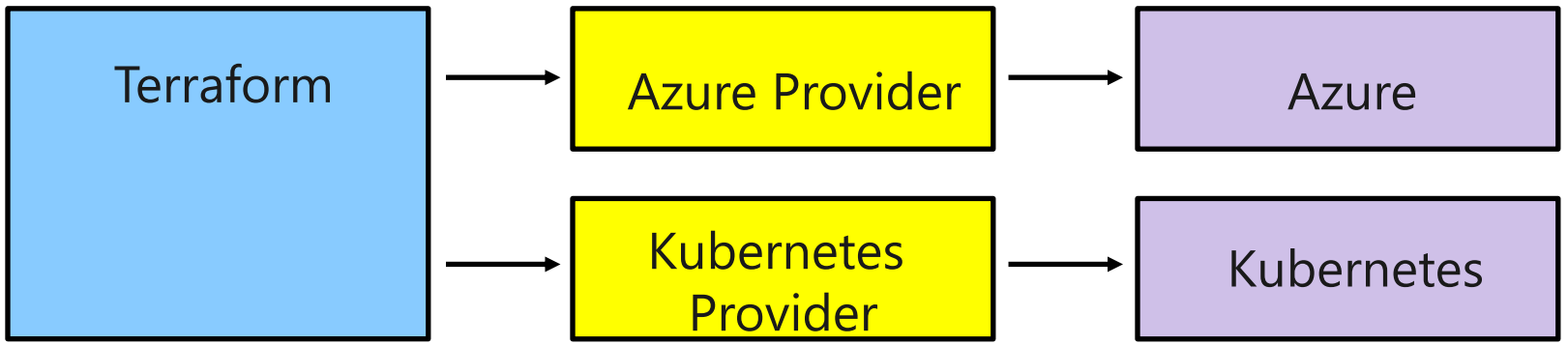
Resource Group

Storage Account

Providers

What is a Terraform provider?

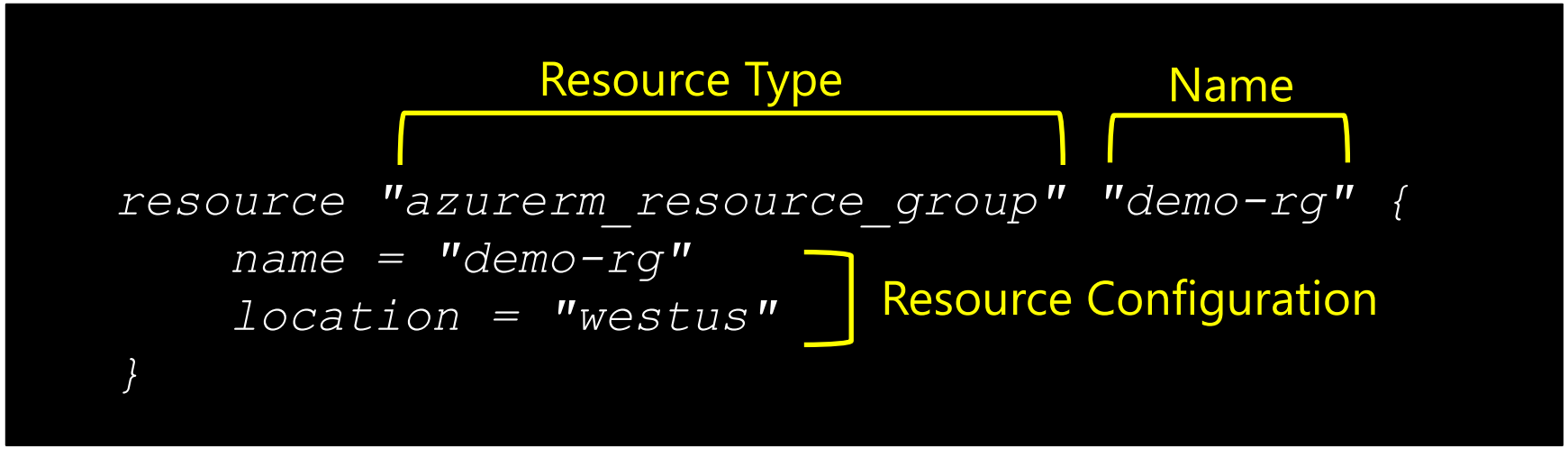
- Terraform ‘extensions’ for deploying resources
- Manages cloud / endpoint specific API interactions
- Available for major clouds and other platforms
- Hand authored (azurerm)



Basic resource creation

Deployment foundations.

- Resource Type: required provider
- Name: internal name
- Configuration: deployment details



Automation, orchestration & Configuration Management Tools

Once we have created the infrastructure, we need to do something with it

- Install Corporate standards
- Establish Authentication / Domain membership
- Install SAP prerequisites
- Install SAP software
- Install other supporting software
- Maintain ongoing configuration
- Perform patches, updates, etc

Automation, orchestration & Configuration Management Tools

Wide Variety of tools available:

- Custom Script Extension & shell script
- Puppet, Chef, Ansible
- Azure DSC
- Etc.

Checklist

Resource group has been setup appropriately
RBAC access for each resource group is setup
System deployed in appropriate region
Ensure resource locks are setup
Compute size chosen for SAP application is per certified per SAP note 1928533
Right choice of OS is made for M Series (SUSE 12 SP3 or RHEL 7.4)
Storage configured in the VM and striped appropriately
Read cache enabled for database data disks
Write acclerator enabled for HANA log disks
Accelerated networking is enabled
Network security group and firewall is setup appropriately (conisder RFC connections)
Subscription mangement server is setup
OS patches are installed and upto date
saptune parameters is applied for HANA DB VM's
OS parameters per SAP note is configured (parameters differs for every release and service pack of operating system)
Ensure ER gateway is configured
Ensure SAP systems are deployed in the appropriate VNETs
configure VNET peering if required
Ensure cluster parameters are in place
Perform failover testing after cluster configuration
Ensure appropriate I/O fencing mechanisms are configured
Ensure NTP is setup
Time is synchronized across application and databases VM's
Ensure appropriate mountpoints are setup
for SAP HANA parameters refer to SAP Note 1969700 - SQL Statement Collection for SAP HANA
Refer SAP Note: 2186744 - FAQ: SAP HANA Parameters

SAP License – SLICENSE
Initial consistency check - SICK
Application instances – SM51
Operation mode check – RZ03
Web Dispatcher
Transport Management System
SMQ1/SMQ2
email (SCOT)
Check LC10, DB59, and connect to Live Cache from SAP
Check all the important RFC destinations and critical interfaces
Enqueue Replication incase of HA
SMD agents are installed
Ensure connectivity to solution manager
DBACOCKPIT: Main database analysis tasks, SQL editor needs to be available
AL11: Directory access
RSPFPAR: SAP profile parameters
SE11: ABAP dictionary
SE37: Execution of function modules (DB_EXECUTE_SQL, …)
SE38 / SA38: Execution of reports
SM21: SAP syslog (logs from all instances)
SM49: Execution of logical commands
SM50: Instance specific SAP work process overview
SM51: SAP instance overview
SM66: Global SAP work process overview
SNRO: Number range information
ST02: SAP buffers and memory
ST03 / ST03N: SAP workload analysis
ST05: Execution plans, SQL trace
ST06: Operating system analysis
ST10: SAP table buffer analysis
ST22: SAP short dumps
STAD: SAP statistical records

Infrastructure as Code + key concepts

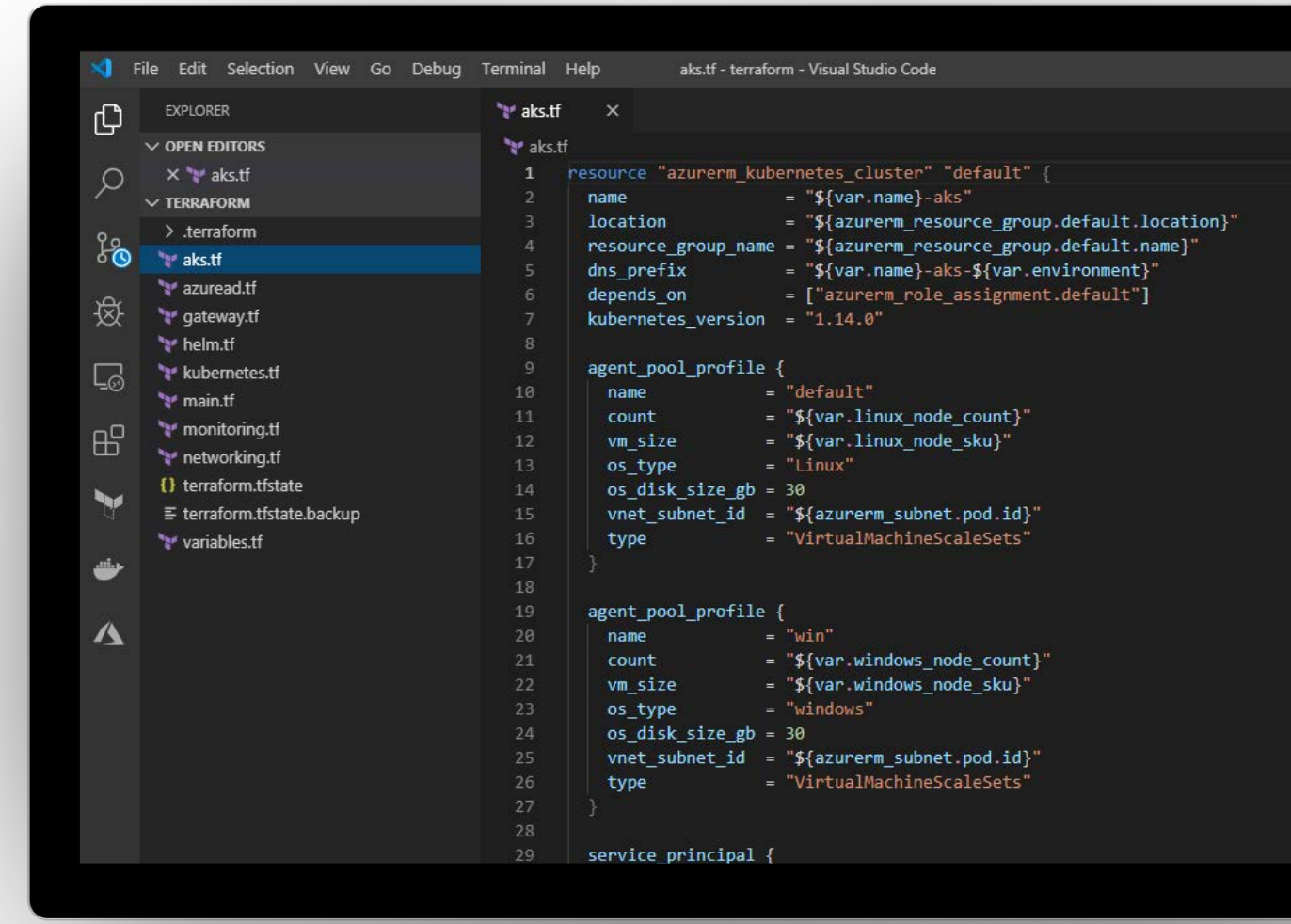
Infrastructure as code

Stand up environments in the fastest means possible.

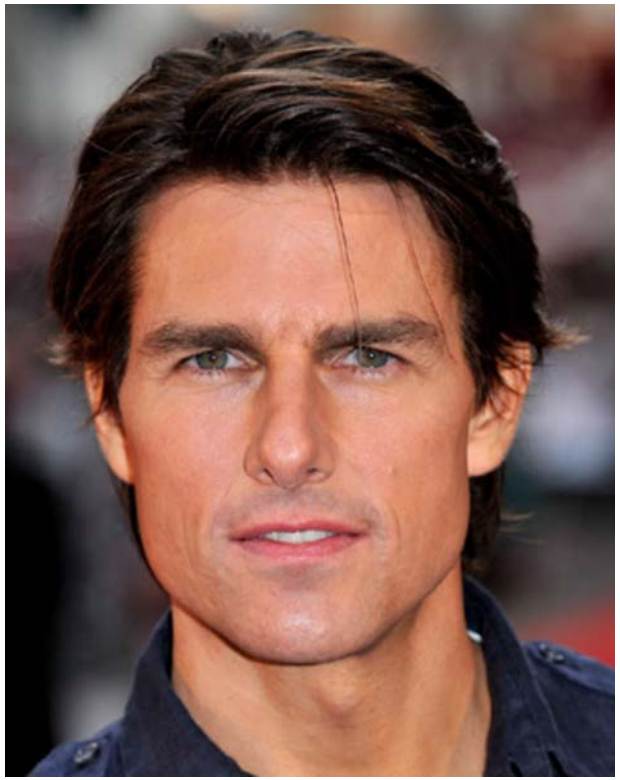
Reduce human errors and increase reliably.

Improve environment visibility and improve developer efficiency

Store your infrastructure definitions alongside your application code.

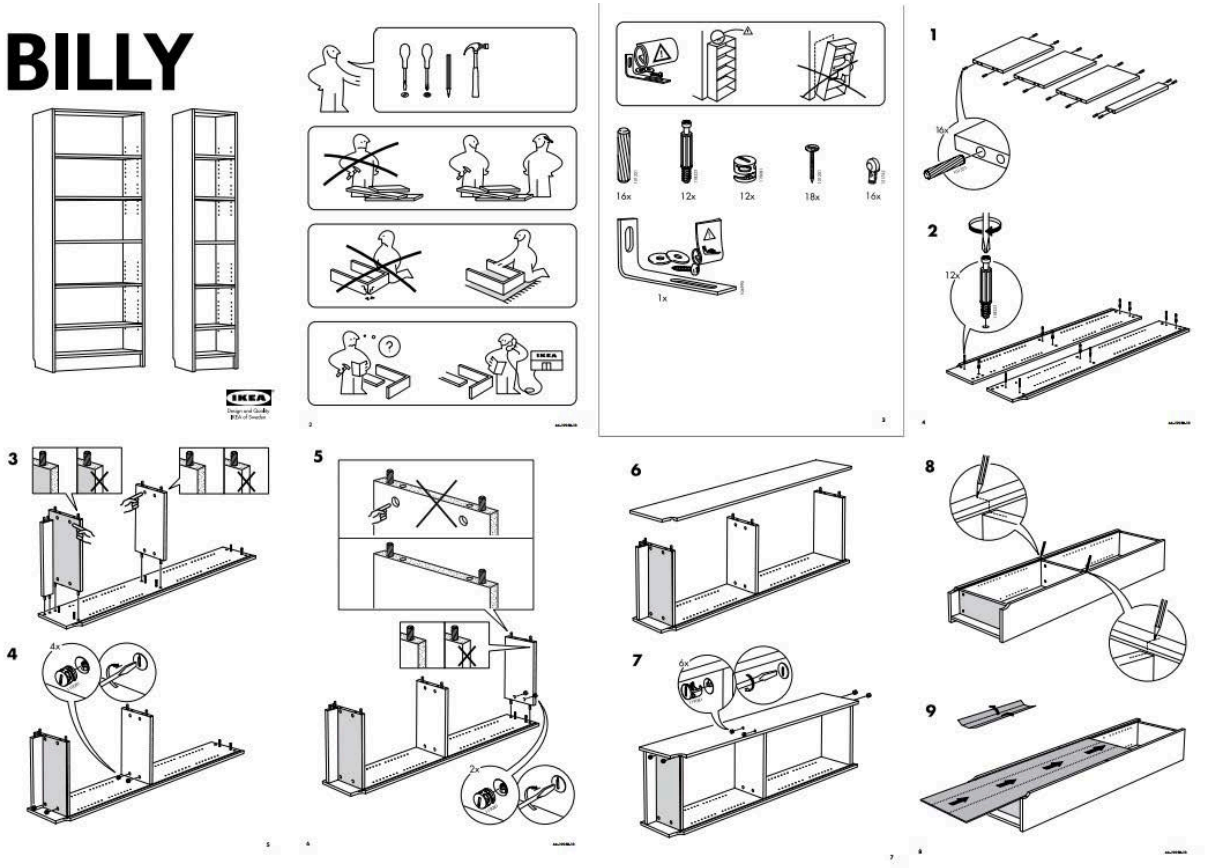


Pet vs Cattle



Mutable vs Immutable

Frying vs Baking



Imperative vs Declarative

Snowflake vs Phoenix

Azure subscription

GitOps

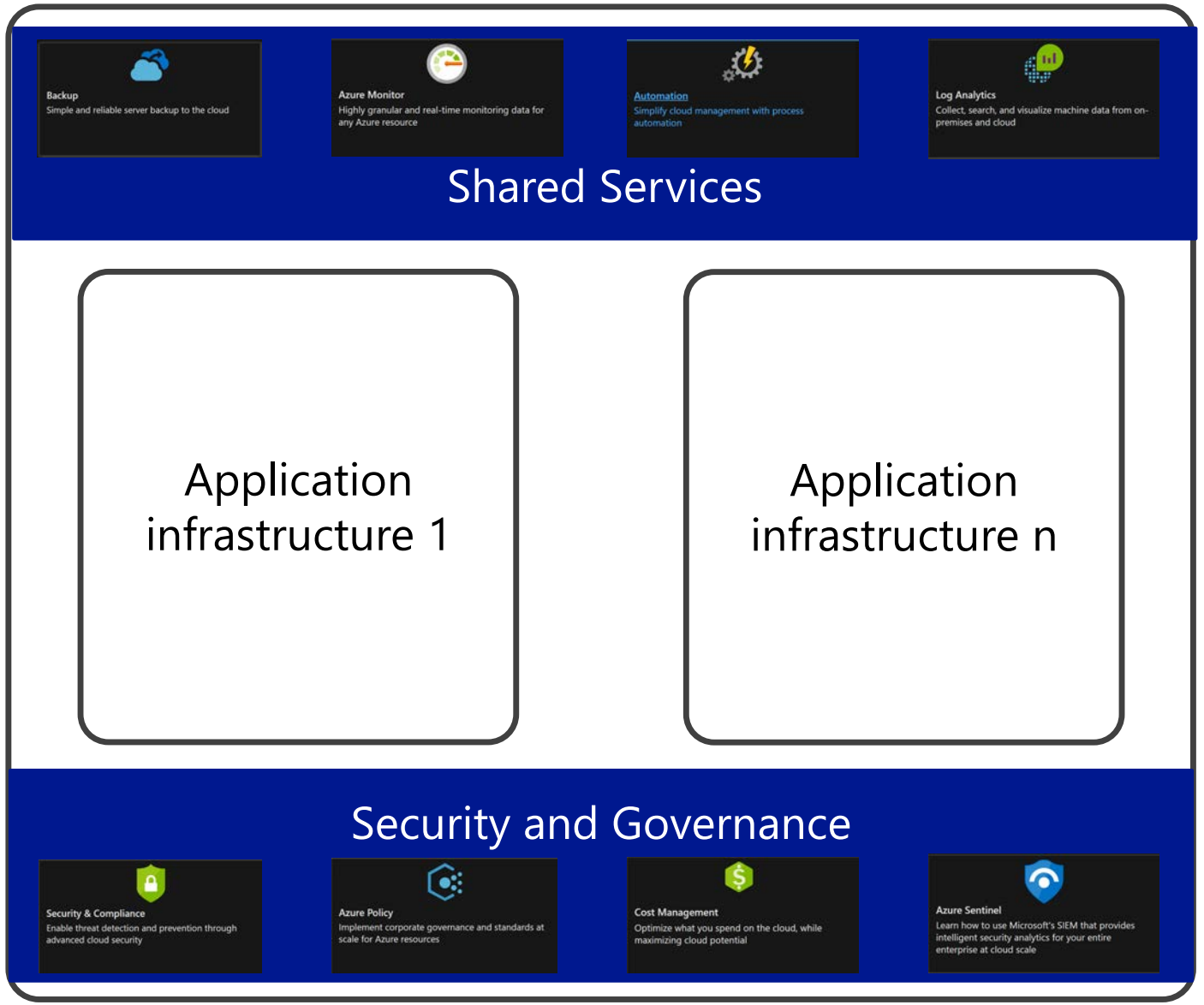
config registry | GIT repos | build agents | release agents

Remote state management

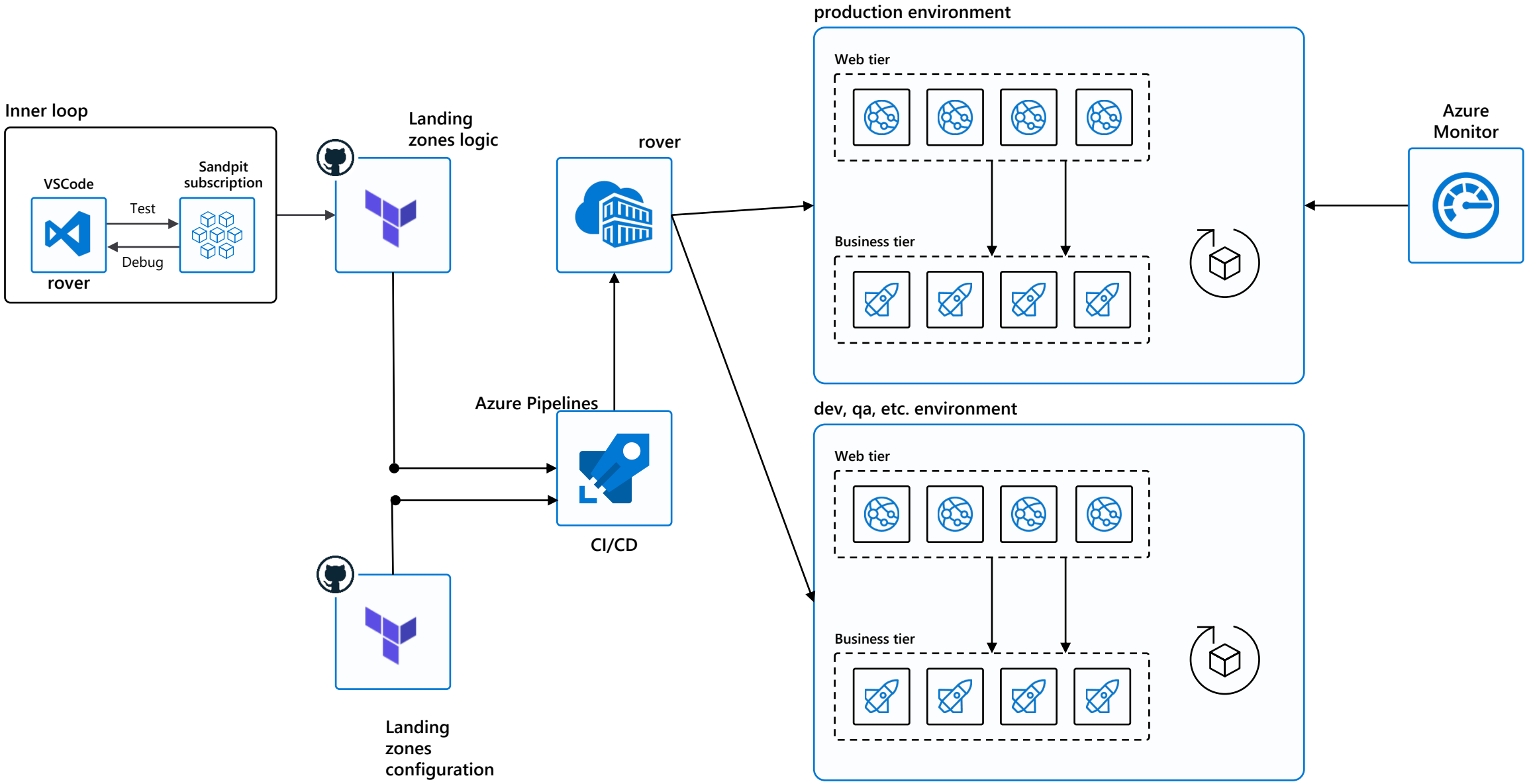
storage account | VNET | keyvault | identity

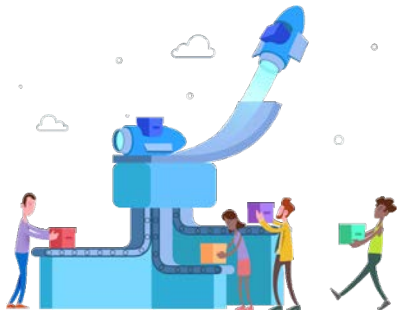
Launchpad

Azure subscription with landing zones



Operating IaC - CI/CD Pipeline

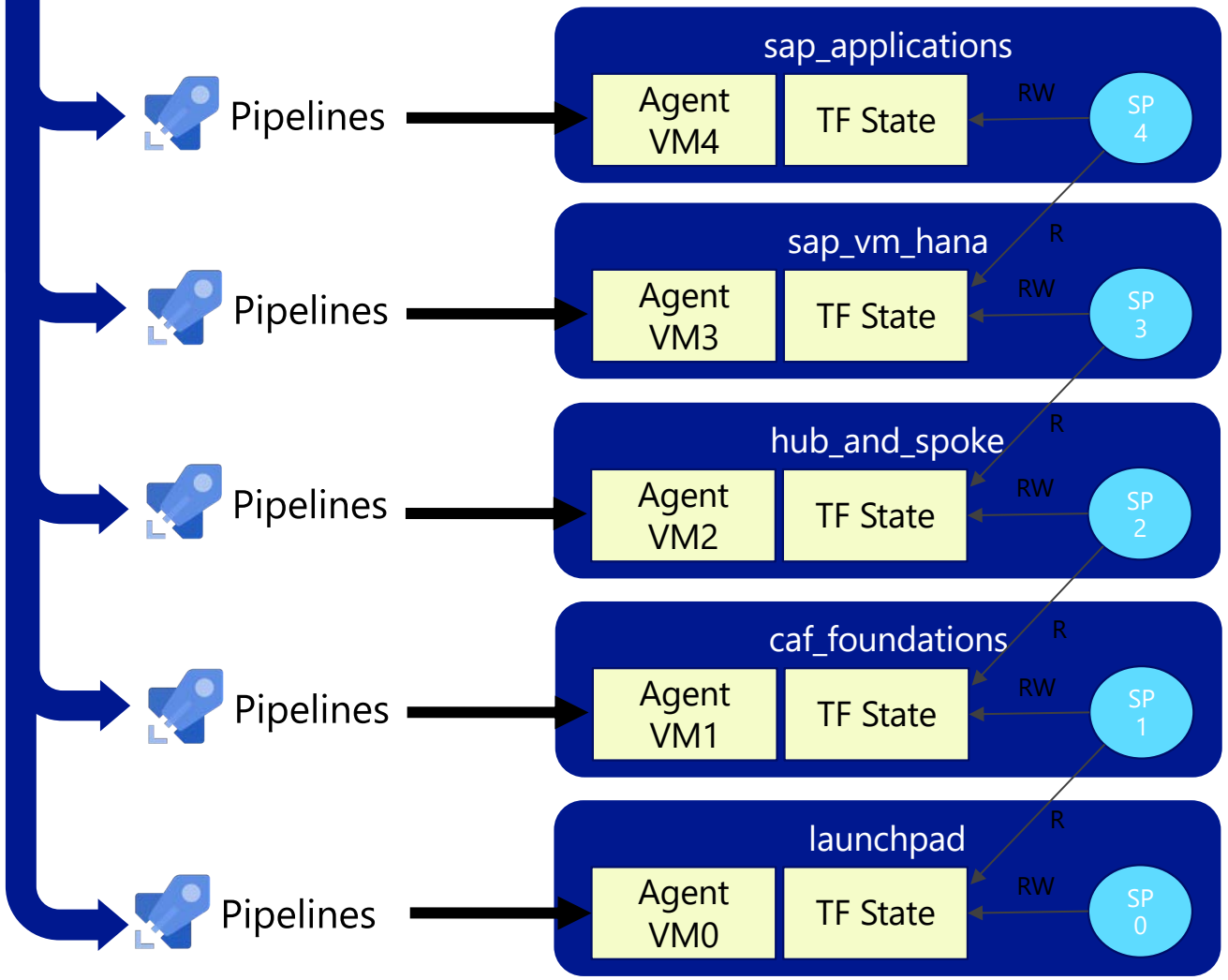




Azure Pipelines

Continuously build, test, and deploy to any platform and cloud

Landing zones structuring example



Service Principal privilege reduction

Identity segmentation

Manage the deployment of an **application itself / ML model** in the application's landing zone e.g (Springboot microservices, dotnet core…)

Manage the deployment of an **application's landing zone** in a spoke environment (e.g AKS Cluster + WAF)

Manage the **network hub and spokes** and **shared services** of each environments, as described in the design document (backup, DR, Azure monitor, patch management…)

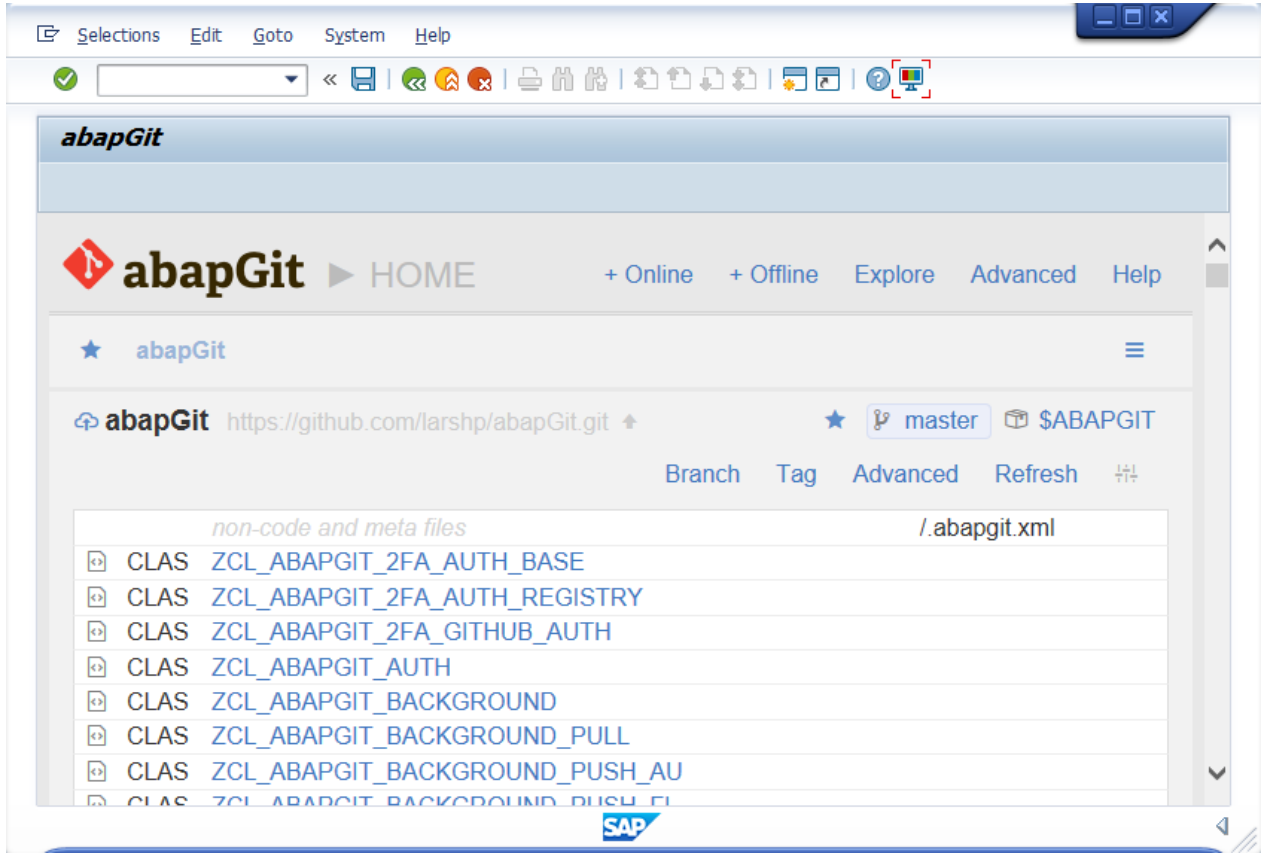
Manage the **security and compliance** (RBAC, Policies, OMS monitoring, shared security services, including event hub collector for SIEM, preventive and reactive controls…)

Transition from manual to automation. Create the subscriptions (for level 0 to level 4) + terraform state repository, privileged access workstation, service principals

Automating SAP deployment

abapGit – DevOps for SAP

- Created by Lars Hvam – Has many contributors
- Open Source project enabling SAP integration with Git
- Works on ABAP Package Level
- User Exits allows Fine-Tuning
- Single Sign-on using Azure Active Directory
- Integration with Azure DevOps



https://github.com/Azure/sap-hana

SAP Automation

Automated SAP Deployments in Azure Cloud

Master Branch's status: Azure Pipelines never built

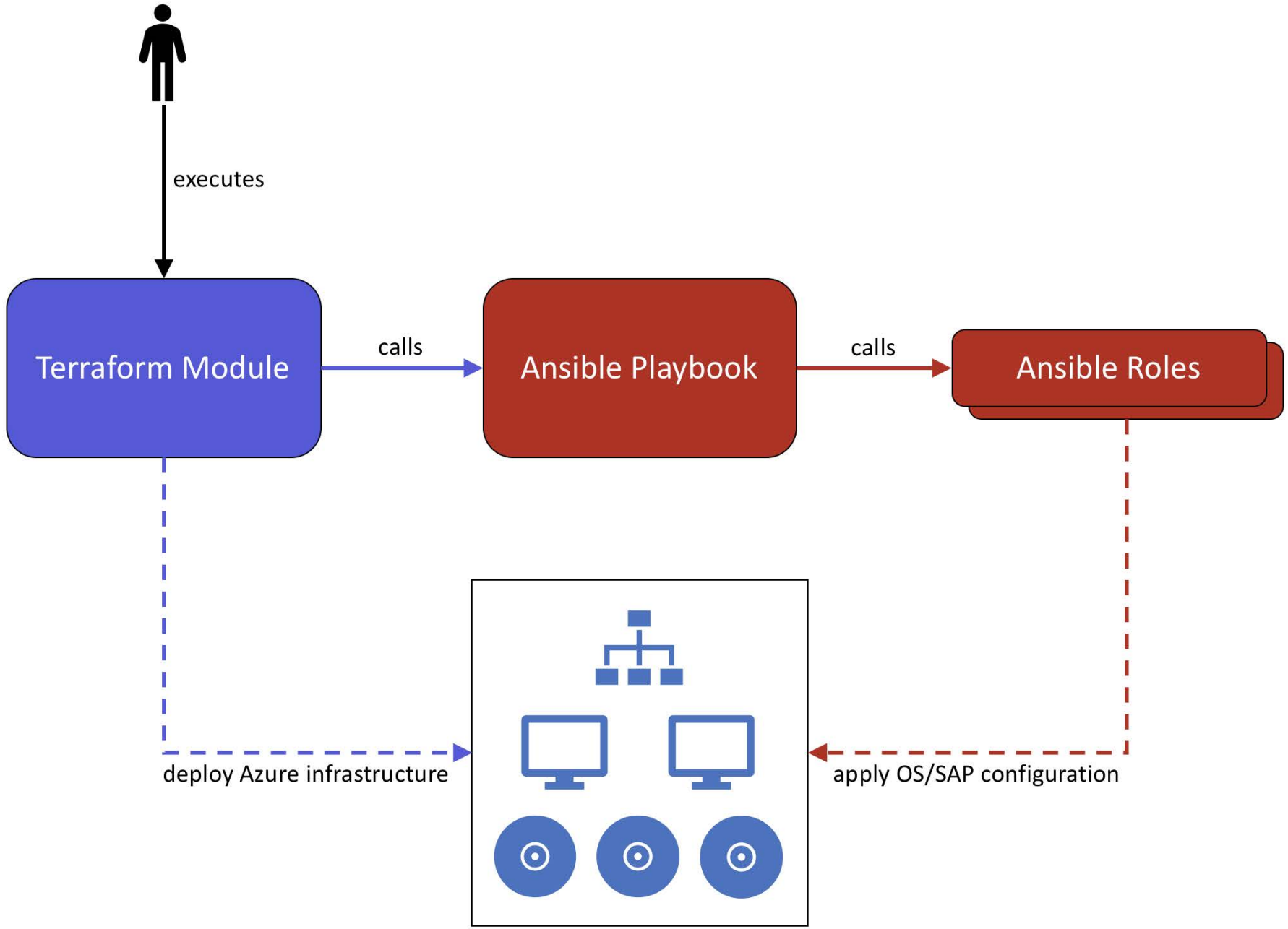
This repository contains code that can be used to automatically deploy SAP landscapes in the Azure Cloud.

The templates are split into:

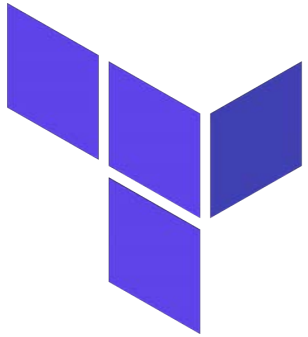
- **Terraform modules** which deploy the infrastructure components (such as VMs, network, storage) in Azure.
- **Ansible playbooks** which run different roles to configure and VMs and install SAP HANA and required applications on the already deployed infrastructure.

A new approach to automating SAP

- Terraform modules which deploy the infrastructure components (such as VMs, network, storage) in Azure and then call the:
- Ansible playbook which call different:
- Ansible roles to install and configure OS and SAP applications on the deployed infrastructure in Azure.


executes
Terraform Module
calls
Ansible Playbook
calls
Ansible Roles
deploy Azure infrastructure
apply OS/SAP configuration

Supported Scenarios Available



- HANA Scale-Up Stand Alone
- HANA with High Availability

Usage

- A typical deployment lifecycle will require the following steps:

1. **Initialize the Deployment Workspace**
2. **Adjusting the templates**
3. **Running Terraform deployment**
4. **Running Ansible Playbook**
5. **Deleting the deployment** (optional)
6. *(**Note**: There are some script under sap-hana/util would help if you are using Linux based workstation)*

What will be deployed

Common Infrastructure

Resource Group

Management VNet

Management SubNet and Network Security Group (NSG)

SAP VNet

Storage Account for SAP Media downloads

Proximity Placement Group

iSCSI SubNet, NSG and Network Security Rules (NSR)

iSCSI VMs

HANA Database

Administration SubNet, NSG, and NSR

Database SubNet, NSG, and NSR

HDB Availability Set and Load Balancer

HDB VMs with Data Disks

Application Tier

Application Tier SubNet, NSG, and NSR

SAP Central Services (SCS) Load Balancer and Rules

Web Dispatcher Load Balancer and Rules

Application, SCS, and Web Dispatcher Availability Sets

Application VMs (based on a count)

SCS VMs (Stand Alone or 2 VMs for High Availability)

Web Dispatcher VMs (based on a count)

Jumpboxes

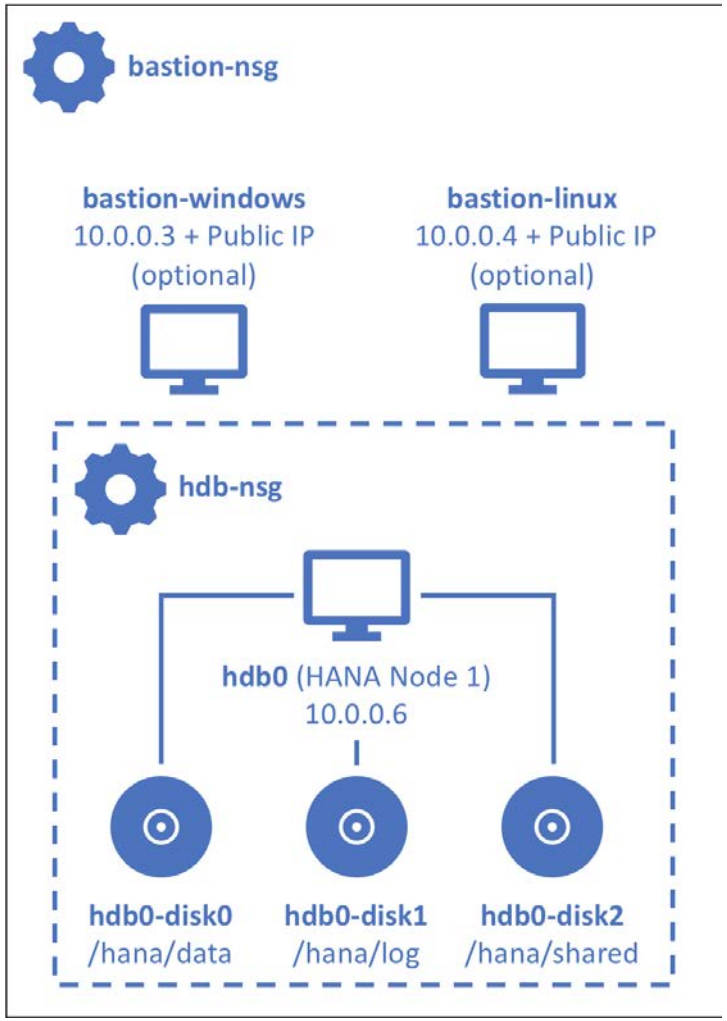
Linux based VM Jumpbox

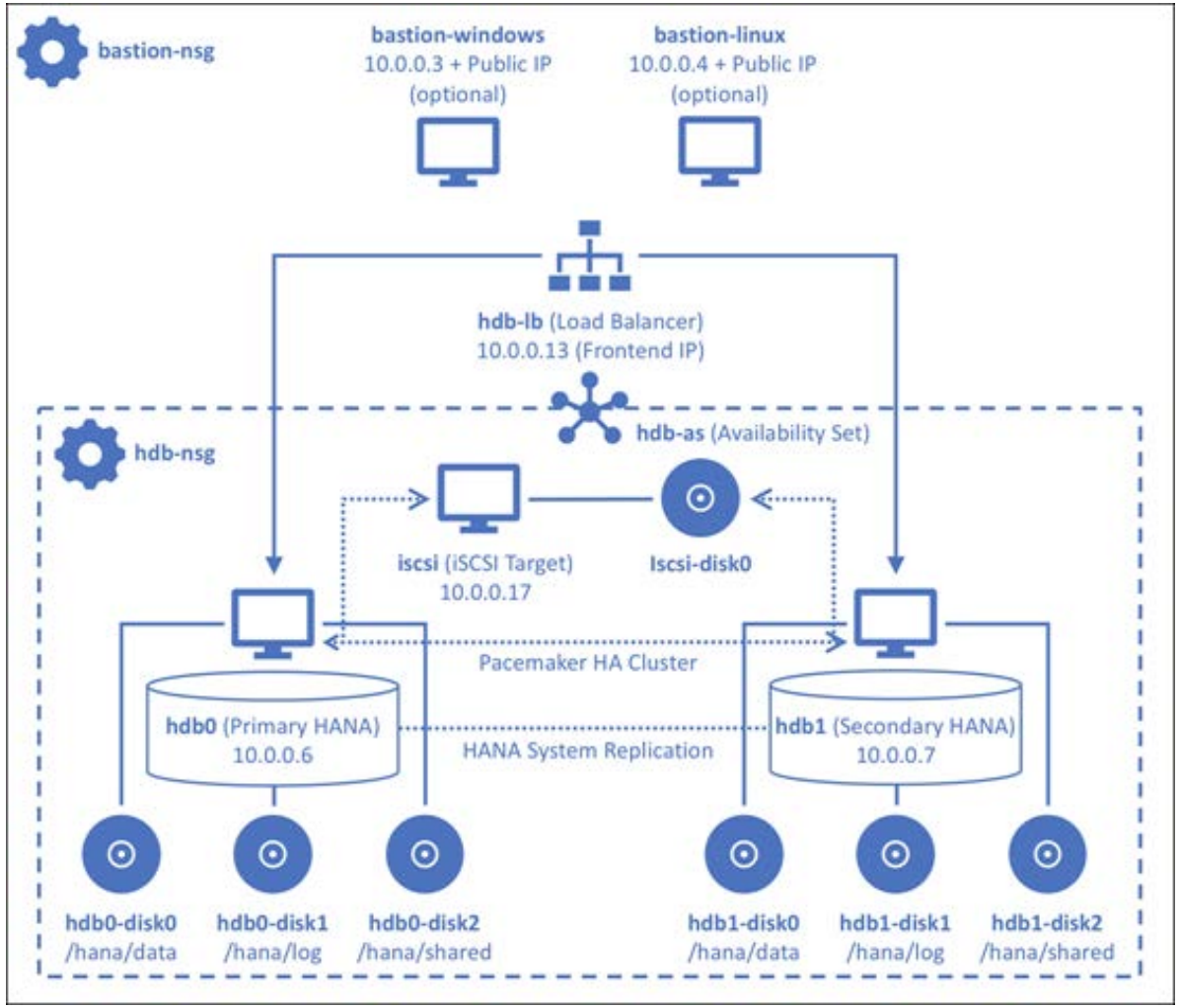
Windows based VM Jumpbox

Run Time Instance VM for Ansible Configuration

*(**Note:** The Run Time Instance is a Linux jumpbox configured with the Ansible component for configuring the other Virtual machines deployed)*

Scenarios


bastion-nsg
bastion-windows
10.0.0.3 + Public IP
(optional)
bastion-linux
10.0.0.4 + Public IP
(optional)
hdb-nsg
hdb0 (HANA Node 1)
10.0.0.6
hdb0-disk0
/hana/data
hdb0-disk1
/hana/log
hdb0-disk2
/hana/shared


bastion-nsg
bastion-windows
10.0.0.3 + Public IP
(optional)
bastion-linux
10.0.0.4 + Public IP
(optional)
hdb-lb (Load Balancer)
10.0.0.13 (Frontend IP)
hdb-as (Availability Set)
hdb-nsg
iscsi (iSCSI Target)
10.0.0.17
Iscsi-disk0
Pacemaker HA Cluster
hdb0 (Primary HANA)
10.0.0.6
HANA System Replication
hdb1 (Secondary HANA)
10.0.0.7
hdb0-disk0
/hana/data
hdb0-disk1
/hana/log
hdb0-disk2
/hana/shared
hdb1-disk0
/hana/data
hdb1-disk1
/hana/log
hdb1-disk2
/hana/shared

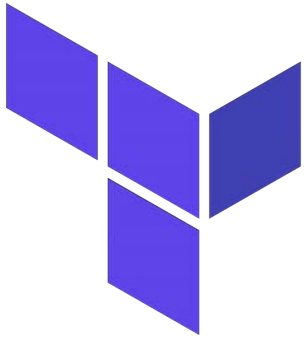
Step 1 – prepare the environment

- Install Terraform
- Install Azure CLI
- Install Ansible

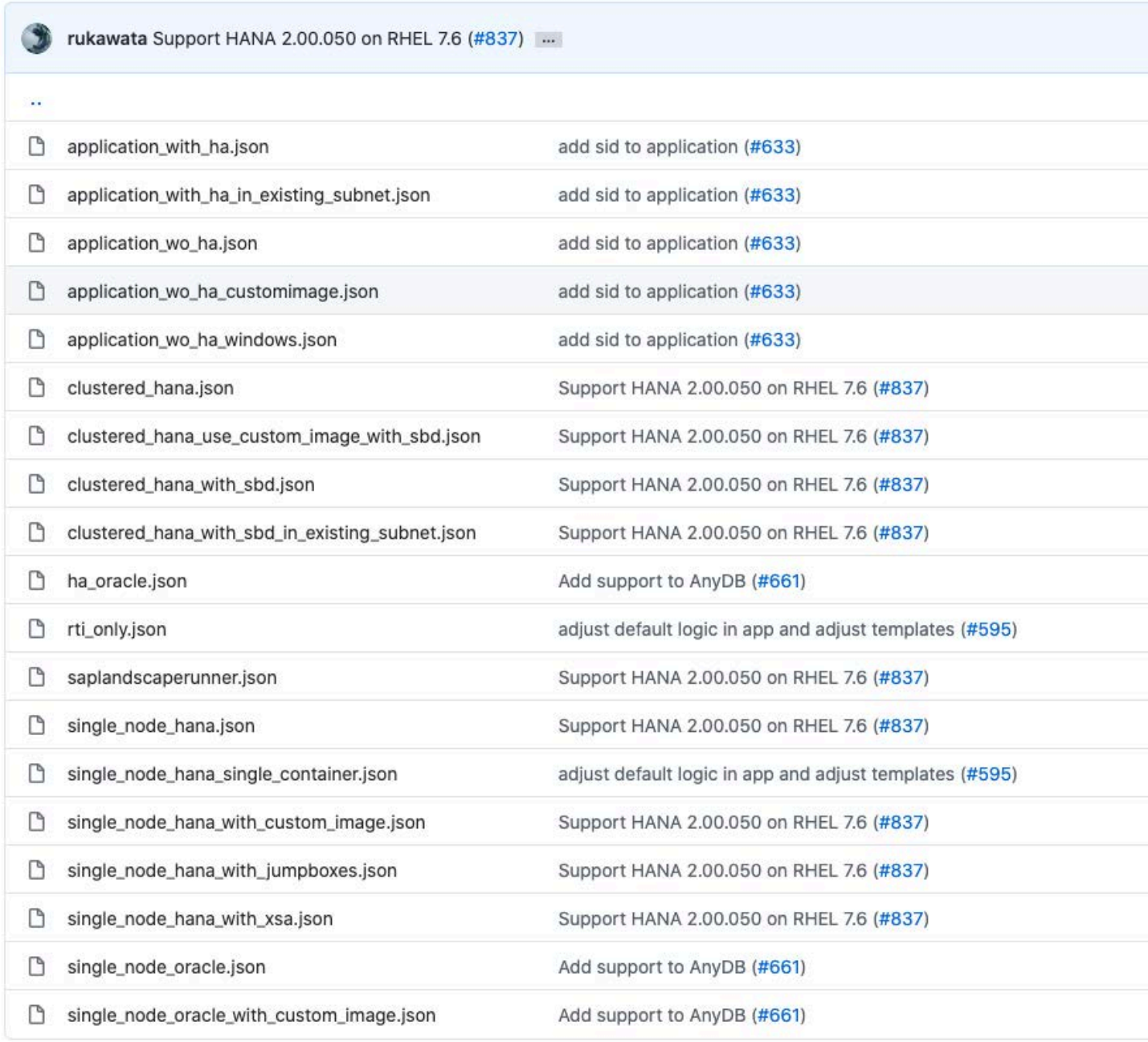
- Prepare your subscription: Service Principal or MSI
  - `az ad sp create-for-rbac --name <service-principal-name>`
  - Assign Contributor role for the required scope

- Clone the repo and create a Terraform Workspace
- Configure Terraform Azure RM provider

- export ARM_SUBSCRIPTION_ID=${subscription_id}
- export ARM_TENANT_ID=${tenant_id}
- export ARM_CLIENT_ID=${client_id}
- export ARM_CLIENT_SECRET=${client_secret}



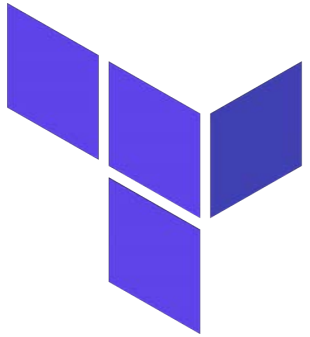
Step 2 - Choose your scenario



rukawata Support HANA 2.00.050 on RHEL 7.6 (#837) ...

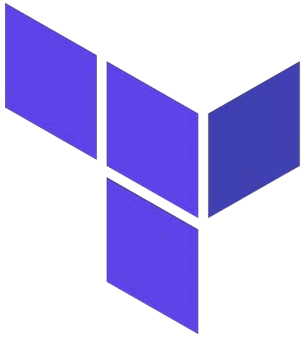
..	
application_with_ha.json	add sid to application (#633)
application_with_ha_in_existing_subnet.json	add sid to application (#633)
application_wo_ha.json	add sid to application (#633)
application_wo_ha_customimage.json	add sid to application (#633)
application_wo_ha_windows.json	add sid to application (#633)
clustered_hana.json	Support HANA 2.00.050 on RHEL 7.6 (#837)
clustered_hana_use_custom_image_with_sbd.json	Support HANA 2.00.050 on RHEL 7.6 (#837)
clustered_hana_with_sbd.json	Support HANA 2.00.050 on RHEL 7.6 (#837)
clustered_hana_with_sbd_in_existing_subnet.json	Support HANA 2.00.050 on RHEL 7.6 (#837)
ha_oracle.json	Add support to AnyDB (#661)
rti_only.json	adjust default logic in app and adjust templates (#595)
saplandscaperunner.json	Support HANA 2.00.050 on RHEL 7.6 (#837)
single_node_hana.json	Support HANA 2.00.050 on RHEL 7.6 (#837)
single_node_hana_single_container.json	adjust default logic in app and adjust templates (#595)
single_node_hana_with_custom_image.json	Support HANA 2.00.050 on RHEL 7.6 (#837)
single_node_hana_with_jumpboxes.json	Support HANA 2.00.050 on RHEL 7.6 (#837)
single_node_hana_with_xsa.json	Support HANA 2.00.050 on RHEL 7.6 (#837)
single_node_oracle.json	Add support to AnyDB (#661)
single_node_oracle_with_custom_image.json	Add support to AnyDB (#661)

Step 3 – Update the deployment template



- The SAP environments deployed are configured by JSON input files. These configuration files provide a high degree of customization for the user

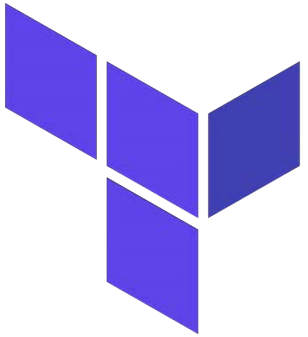
Initialize Terraform



terraform init -var-file=../sap-hana/deploy/template_samples/single_node_hana.json ../sap-hana/deploy/terraform

The terraform init command is used to initialize a working directory containing Terraform configuration files. This is the first command that should be run after writing a new Terraform configuration or cloning an existing one from version control. It is safe to run this command multiple times.

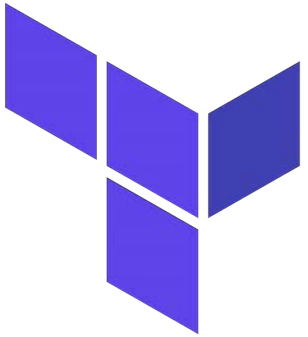
Preview deployment



terraform plan -var-file=../sap-hana/deploy/template_samples/single_node_hana.json ../sap-hana/deploy/terraform

The terraform plan command is used to create an execution plan. Terraform performs a refresh, unless explicitly disabled, and then determines what actions are necessary to achieve the desired state specified in the configuration files.

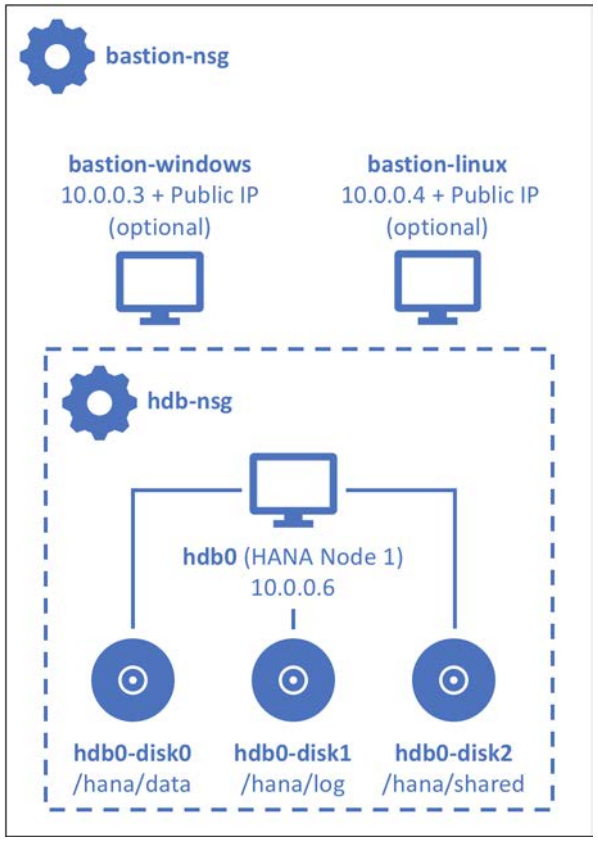
Apply deployment



terraform apply -var-file=../sap-hana/deploy/template_samples/single_node_hana.json ../sap-hana/deploy/terraform

The terraform apply command is used to apply the changes required to reach the desired state of the configuration, or the pre-determined set of actions generated by a terraform plan execution plan.

Infrastructure deployment completed



hdb1-0	Virtual machine
hdb1-0-admin-nic	Network interface
hdb1-0-backup-0	Disk
hdb1-0-data-0	Disk
hdb1-0-db-nic	Network interface
hdb1-0-log-0	Disk
hdb1-0-osdisk	Disk
hdb1-0-sap-0	Disk
hdb1-0-shared-0	Disk
HN1_hdb-alb	Load balancer
HN1_hdb-avset	Availability set
nsg-admin	Network security group
nsg-db	Network security group
nsg-mgmt	Network security group
rti	Virtual machine
rti-nic1	Network interface
rti-osdisk	Disk
rti-public-ip	Public IP address
sabootdiagc96a3f18	Storage account
sapbitsc96a3f18	Storage account
test-ppg	Proximity placement group
vnet-mgmt	Virtual network
vnet-sap	Virtual network

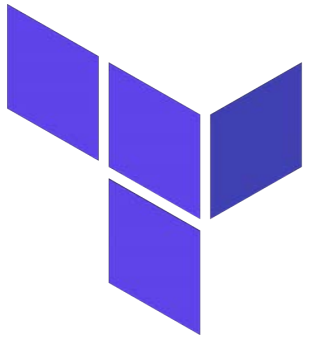
```
Outputs:

ansible-jumpbox-public-ip-address = 52.170.212.
ansible-jumpbox-username = azureadm
```

Configuring SAP with Ansible

- The infrastructure has been deployed with Terraform
- We will use Ansible to configure SAP

- Note: you can automate the entire process by specifying

```
"options": {
        "enable_secure_transfer": true,
        "ansible_execution": true,
        "enable_prometheus": true
},
```



RTI + Jumpbox configuration

```
module.jumpbox.null_resource.prepare-rti (remote-exec): Selecting previously unselected package dpkg-dev.
module.jumpbox.null_resource.prepare-rti (remote-exec): Preparing to unpack .../36-dpkg-dev_1.19.0.5ubuntu2.3_a
Progress: [ 35%] [######............] ti (remote-exec): Unpacking dpkg-dev (1.19.0.5ubuntu2.3) ...
module.jumpbox.null_resource.prepare-rti (remote-exec): Selecting previously unselected package build-essential.
module.jumpbox.null_resource.prepare-rti (remote-exec): Preparing to unpack .../37-build-essential_12.4ubuntu1_a
Progress: [ 36%] [######............] ti (remote-exec): Unpacking build-essential (12.4ubuntu1) ...
module.jumpbox.null_resource.prepare-rti (remote-exec): Selecting previously unselected package python3-lib2to3.
module.jumpbox.null_resource.prepare-rti (remote-exec): Preparing to unpack .../38-python3-lib2to3_3.6.9-1~18.04
module.jumpbox.null_resource.prepare-rti (remote-exec): Unpacking python3-lib2to3 (3.6.9-1~18.04) ...
Progress: [ 37%] [######............] ti (remote-exec): Selecting previously unselected package python3-distutil
module.jumpbox.null_resource.prepare-rti (remote-exec): Preparing to unpack .../39-python3-distutils_3.6.9-1~18.
module.jumpbox.null_resource.prepare-rti (remote-exec): Unpacking python3-distutils (3.6.9-1~18.04) ...
Progress: [ 38%] [######............] ti (remote-exec): Selecting previously unselected package dh-python.
module.jumpbox.null_resource.prepare-rti (remote-exec): Preparing to unpack .../40-dh-python_3.20180325ubuntu2_a
module.jumpbox.null_resource.prepare-rti (remote-exec): Unpacking dh-python (3.20180325ubuntu2) ...
Progress: [ 39%] [#######...........] ti (remote-exec): Selecting previously unselected package libfakeroot:amd6
module.jumpbox.null_resource.prepare-rti (remote-exec): Preparing to unpack .../41-libfakeroot_1.22-2ubuntu1_amd
module.jumpbox.null_resource.prepare-rti (remote-exec): Unpacking libfakeroot:amd64 (1.22-2ubuntu1) ...
Progress: [ 40%] [#######...........] ti (remote-exec): Selecting previously unselected package fakeroot.
module.jumpbox.null_resource.prepare-rti (remote-exec): Preparing to unpack .../42-fakeroot_1.22-2ubuntu1_amd64.
module.jumpbox.null_resource.prepare-rti (remote-exec): Unpacking fakeroot (1.22-2ubuntu1) ...
Progress: [ 41%] [#######...........] ti (remote-exec): Selecting previously unselected package libalgorithm-dif
module.jumpbox.null_resource.prepare-rti (remote-exec): Preparing to unpack .../43-libalgorithm-diff-perl_1.19.0
```

Ansible playbook

- **Playbooks** are one of the core features of **Ansible** and tell **Ansible** what to execute. They are like a to-do list for **Ansible** that contains a list of tasks. **Playbooks** contain the steps which the user wants to execute on a particular machine. **Playbooks** are run sequentially.

Connect to the Bastion to run the playbook

```
azureadm@rti:~$ ls
export-clustering-sp-details.sh  hosts  hosts.yml  output.json  sap-hana
azureadm@rti:~$ ansible-playbook -i hosts.yml ~/sap-hana/deploy/ansible/sap_playbook.yml

PLAY [localhost] ***************************************************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************************
ok: [localhost]

TASK [Include output JSON] *****************************************************************************************************************************************************
ok: [localhost]

TASK [Create dictionary with HANA database inforamtion from output.JSON] *******************************************************************************************************
ok: [localhost] => (item={'authentication': {'type': 'key', 'username': 'azureadm'}, 'components': {'hana_database': []}, 'credentials': {'cockpit_admin_pa
a_cluster_password': '', 'os_sapadm_password': 'Help4you', 'os_sidadm_password': 'Help4you', 'xsa_admin_password': 'Manager1'}, 'db_version': '2.00.050', '
 {'instance_number': '00', 'sid': 'HN1'}, 'loadbalancer': {'frontend_ip': '10.1.2.4'}, 'nodes': [{'dbname': 'hdb1-0', 'ip_admin_nic': '10.1.1.10', 'ip_db_n
-sap-12-sp5', 'publisher': 'suse', 'sku': 'gen1', 'source_image_id': ''}, 'platform': 'HANA', 'shine': {'email': 'shinedemo@microsoft.com'}, 'size': 'Demo'

PLAY [hanadbnodes] *************************************************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************************
The authenticity of host '10.1.1.10 (10.1.1.10)' can't be established.
ECDSA key fingerprint is SHA256:SKcF1ndlPU0c+GSjhE461W1heBmTIv8NpF+YHs14YTo.
Are you sure you want to continue connecting (yes/no)?
```

Azure Pipelines

Cloud-hosted pipelines for Linux, Windows and macOS, with unlimited minutes for open source

Any language, any platform, any cloud

Build, test, and deploy Node.js, Python, Java, PHP, Ruby, C/C++, .NET, Android, and iOS apps. Run in parallel on Linux, macOS, and Windows. Deploy to Azure, AWS, GCP or on-premises

Extensible

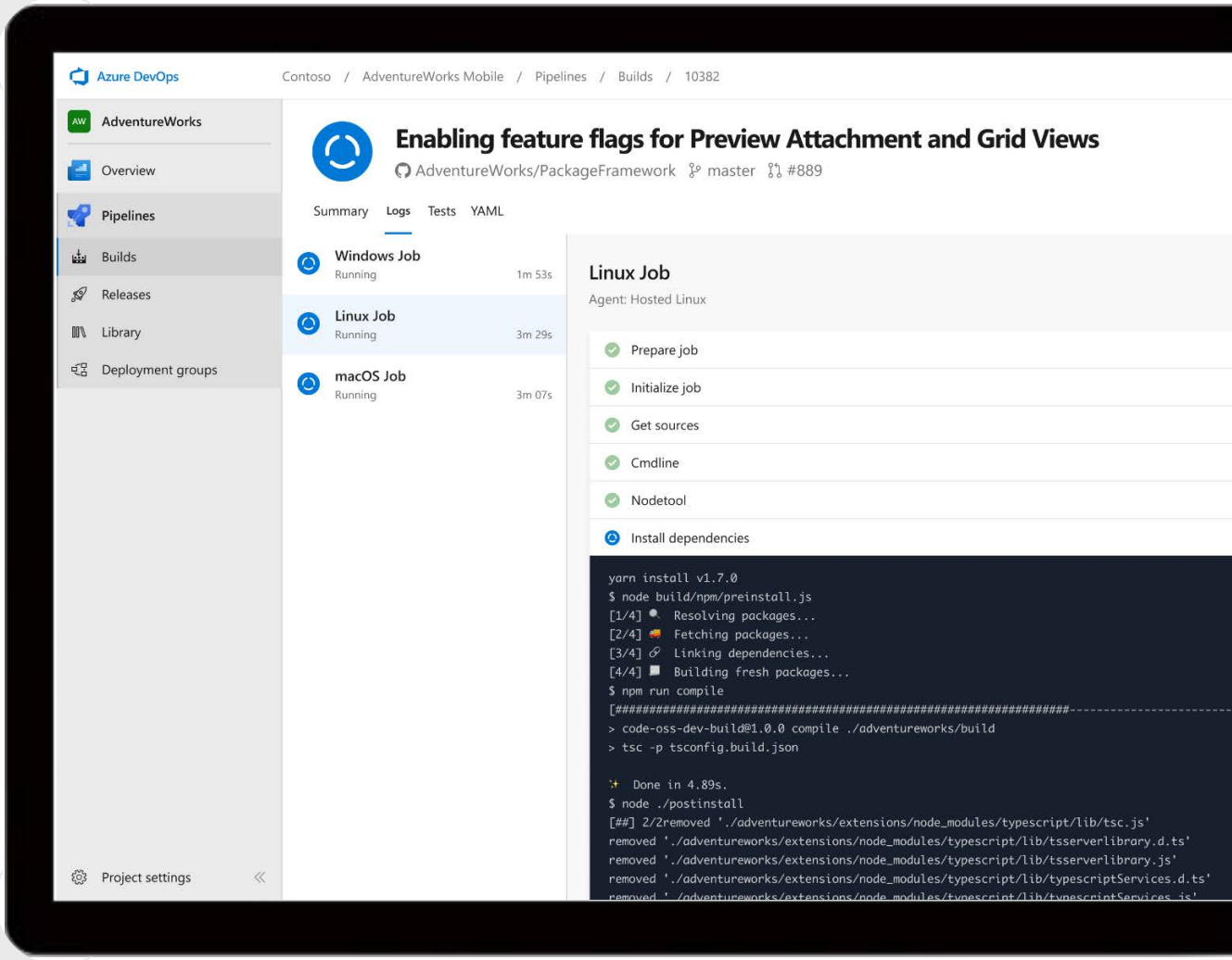
Explore and implement a wide range of community-built build, test, and deployment tasks, along with hundreds of extensions from Slack to SonarCloud. Support for YAML, reporting and more

Containers and Kubernetes

Easily build and push images to container registries like Docker Hub and Azure Container Registry. Deploy containers to individual hosts or Kubernetes.

Best-in-class for open source

Ensure fast continuous integration/continuous delivery (CI/CD) pipelines for every open source project. Get unlimited build minutes for all open source projects with up to 10 free parallel jobs across Linux, macOS and Windows



https://azure.com/pipelines

Azure Pipeline – Concepts

Agents – compute to perform build and release

- Microsoft Managed
- Self-hosted (OnPrem, Azure, Anywhere)

Sources : Azure repo, GitHub, Svn …

Jobs: represents a series of steps that runs in sequence, multiple Jobs can run in Parallel

Tasks: Building block of the pipeline, task is simply a packaged script or procedure that has been abstracted with a set of inputs.

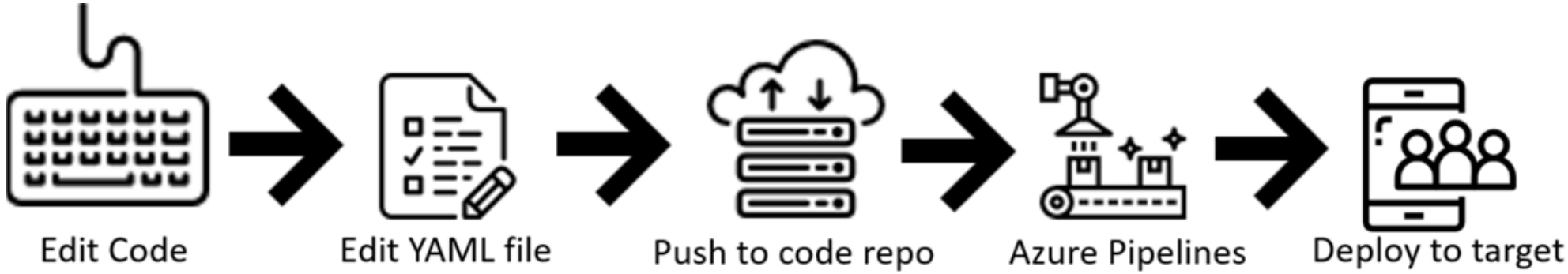
When you run a **job**, all **tasks** run in sequence on an **agent**

Azure Pipeline - Build

- You define you Build pipeline using Agents / jobs / tasks
- Definition can be done using the UI or YAML (recommended)
- YAML can be versioned in Azure Repo
- The Build pipeline generates Artifacts, test results …

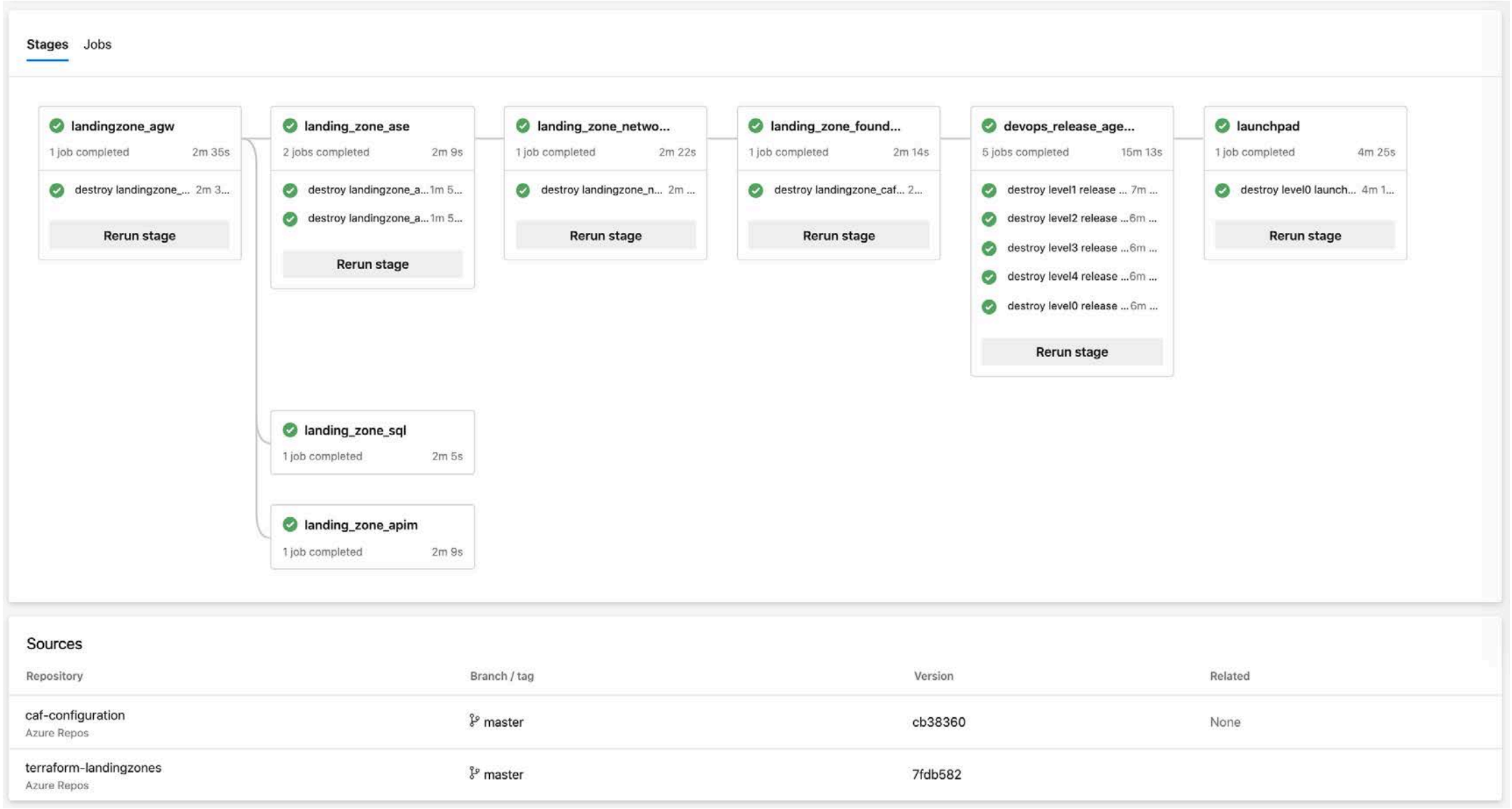
Azure Pipelines and YAML

- Configure your pipelines in a YAML file that exists alongside your code



- Configure Azure Pipelines to use your Git repo
- Edit your azure-pipelines.yml file to define your build
- Push your code to your version control repository
- Your code is now updated, built, tested, and packaged

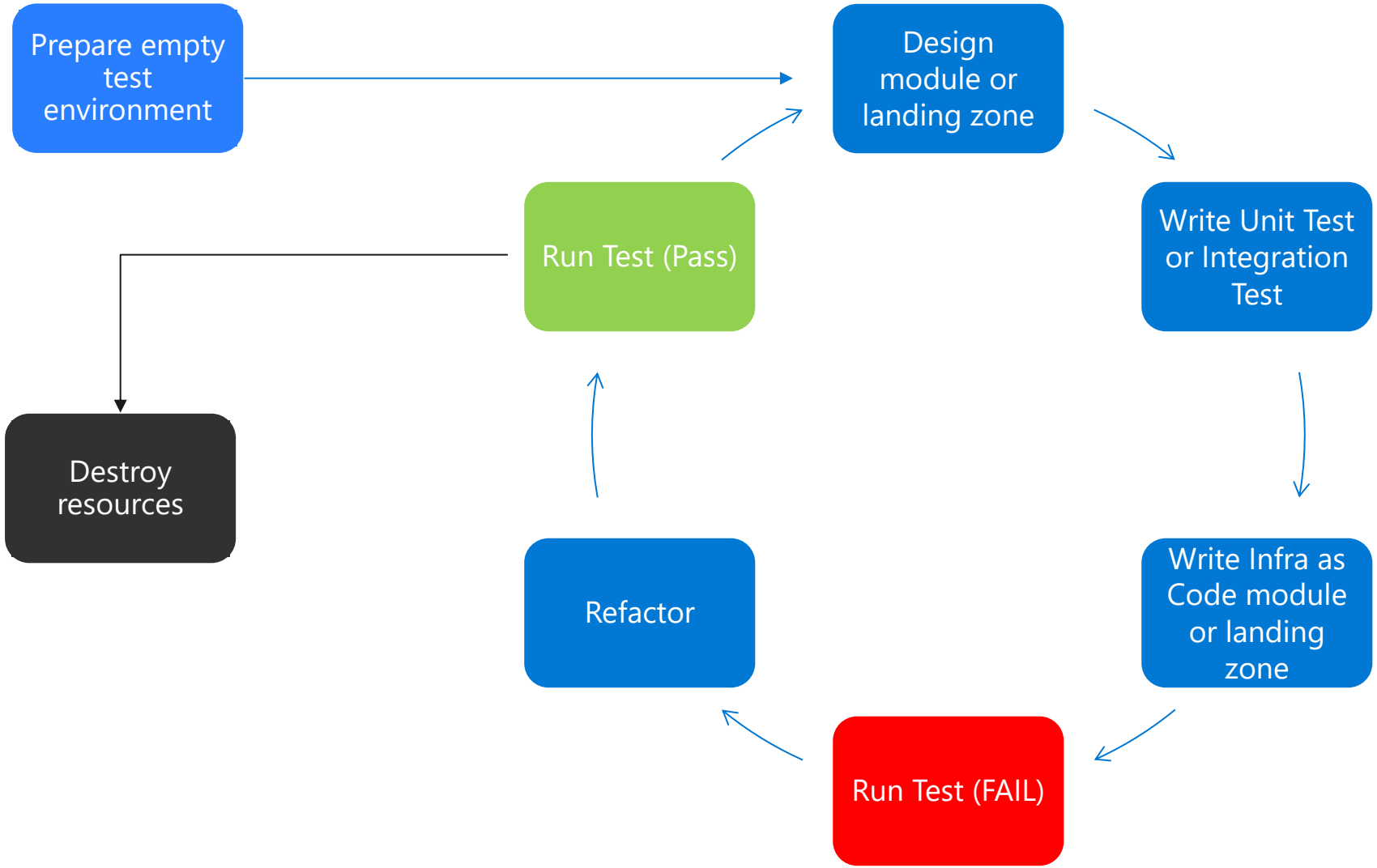
Azure pipeline



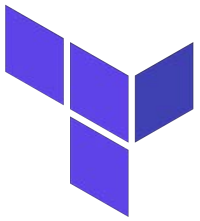
Sources

Repository	Branch / tag	Version	Related
caf-configuration Azure Repos	master	cb38360	None
terraform-landingzones Azure Repos	master	7fdb582	

Testing flow with Infrastructure as Code



Recommended tools



- `terraform show -json`: show a JSON representation of a tfstate file
- `terraform output -json`: extract the value of an output variable from the state file

- `jq` is a lightweight and flexible command-line JSON processor

- Azure SDK and Resource Graph API to retrieve ARM resources as scale using KQL

Unit testing strategy

Objective: validate the configuration of a single component (e.g the terraform code for our module is working as expected)

Stage : terraform `plan` – before any deployment

Methodology :

1. Validate that the plan phase is successful
2. Retrieve the terraform `plan` state file as json
3. Parse the JSON file using your prefered tool or language
4. Extract relevant metadata and compare them to the expected result

Integration testing strategy

Objective: validate the deployment of a module and broader landing zone is successful in the targeted environment

Stage : terraform `apply` – after the deployment in an environment.

Methodology :

1. Validate that the apply phase is successful
2. Retrieve the terraform `output` values as json (e.g resource name)
3. Extract relevant metadata and compare them to the expected result
4. Destroy must be successful (end-to-end test)

Azure Resource Graph

Type of Azure Resource Manager object that we target
(e.g Resources, ResourceContainers)

```
Resources
| where type =~ "microsoft.sql/servers"
| where name contains "svxl-sql-contososg"
| project id, name, type, properties, tags
```

Query filtering

Select the columns to include, rename or drop,
and insert new computed columns.

Azure Resource Graph – integration testing

Written in GoLang and leverages the Azure SDK

`go –t ./…`

1. Uses values from `terraform output` (as env variables)
2. Calls the Azure Resource Graph query
3. Validates the number of results
4. Optional parse the retrieved data (tags, properties)

```go
func TestSQLServer(t *testing.T) {

    t.Parallel()

    serverName := os.Getenv("SQL_SERVER_NAME")
    rgName := os.Getenv("RG_NAME")
    location := os.Getenv("LOCATION")
    kind := os.Getenv("SQL_VERSION")
    env := os.Getenv("ENVIRONMENT")

    var query string = fmt.Sprintf("Resources | where type =~ 'microsoft.sql/servers' |
    subscriptionID := os.Getenv("ARM_SUBSCRIPTION_ID")
        fmt.Println("getting client")
        fmt.Println(query)

    graphClient := resourcegraph.New()
    authorizer, err := auth.NewAuthorizerFromEnvironment()
    if err == nil {
        graphClient.Authorizer = authorizer
    }

    queryRequest := new(resourcegraph.QueryRequest)
    queryRequest.Query = &query
    queryRequest.Subscriptions = &([]string{subscriptionID})
    result, err := graphClient.Resources(context.Background(), *queryRequest)
    if err != nil {
        t.Errorf("%+v\n", err)
    }
    // fmt.Printf(result.Data.(map[string]interface{}).location])

    assert.Equal(t, int64(1), *result.Count)
}
```

Sesssion Takeway

Favor Infra As Code to deploy complex architectures

Automate the deployment process and separate code from configuration (12 factor app)

Implement a DevOps pipeline

Test your SAP deployment

Q&A

Reach out to the team

sap-on-azure-pe-apac@microsoft.com

Feedback

Your feedback is very important for us.

https://aka.ms/SAPAPAC-POE-FEEDBACK

Microsoft

SAP on Azure Enablement

Next Session – Migration and Architecture best practices

Next week - Monday, Oct 26, 2020, 10am SGT

Reach out to the team

sap-on-azure-pe-apac@microsoft.com

Reach out to the team

Ravi Gangampalli

Sajit Nair

Nicolas Yuen

Inseob Kim

sap-on-azure-pe-apac@microsoft.com