

# Integration of Linux on Z/LinuxONE Instances to a CI/CD Pipeline

\*Note: This is not a production-ready configuration for running workloads. This CI/CD sample automation is intended for test/development purposes only until officially promoted. Please refer to the [related documentation](#) for running remote ssh pipelines in Jenkins

## Overview

This repository provides documentatation and a set of sample scripts demonstrating the automatic deployment and management Linux on Z servers and running Jenkins builds on it. The provided here example shows how independent software vendors (ISVs) with the access to IBM LinuxONE ISV Cloud access can integrate IBM Z/LinuxONE builds/tests into their CI/CD pipeline and continuously deliver their application on multiple architectures. It regards common development practices of Agile, DevOps and CI/CD.

## Description

Application development organizations following agile methodology focus on incremental delivery, constant feedback and collaboration. This translates into automation and incremental coding and testing as part of the CI/CD pipeline. IBM Z/LinuxONE server hardware architected for security, resilience and scalability runs popular Linux distros and a large number of opensource products. Such as RHEL, Ubuntu, SUSE, Fedora, Alpine etc. It is suitable for integrating in or even hosting CI/CD pipelines.

A good CI/CD pipeline has to meet speed, availability and accuracy requirements of the agile development. You want the builds and tests be reproducible regardless of the underlying CI/CD agents, OS versions, and infrastructure. Be able to scale the CI/CD in real-time to cover the spikes in the development team demands as well as to support ongoing functionality and codebase increases. For better accuracy, make workflows start from the same clean state and run in an isolated environment. Ensure major part of the CI/CD meets the requirements and follow the best practices.

This repository is a Jenkins pipeline hosted on the ISV's CI/CD server and running a terraform script. The script establishes VPN connection and interacts with the OpenStack APIs of the LinuxONE ISV Cloud. Creates a RHEL instance and executes shell commands via s secure shell - ssh.

Two scenarios of provisioning/deprovisioning Linux on Z servers are covered:

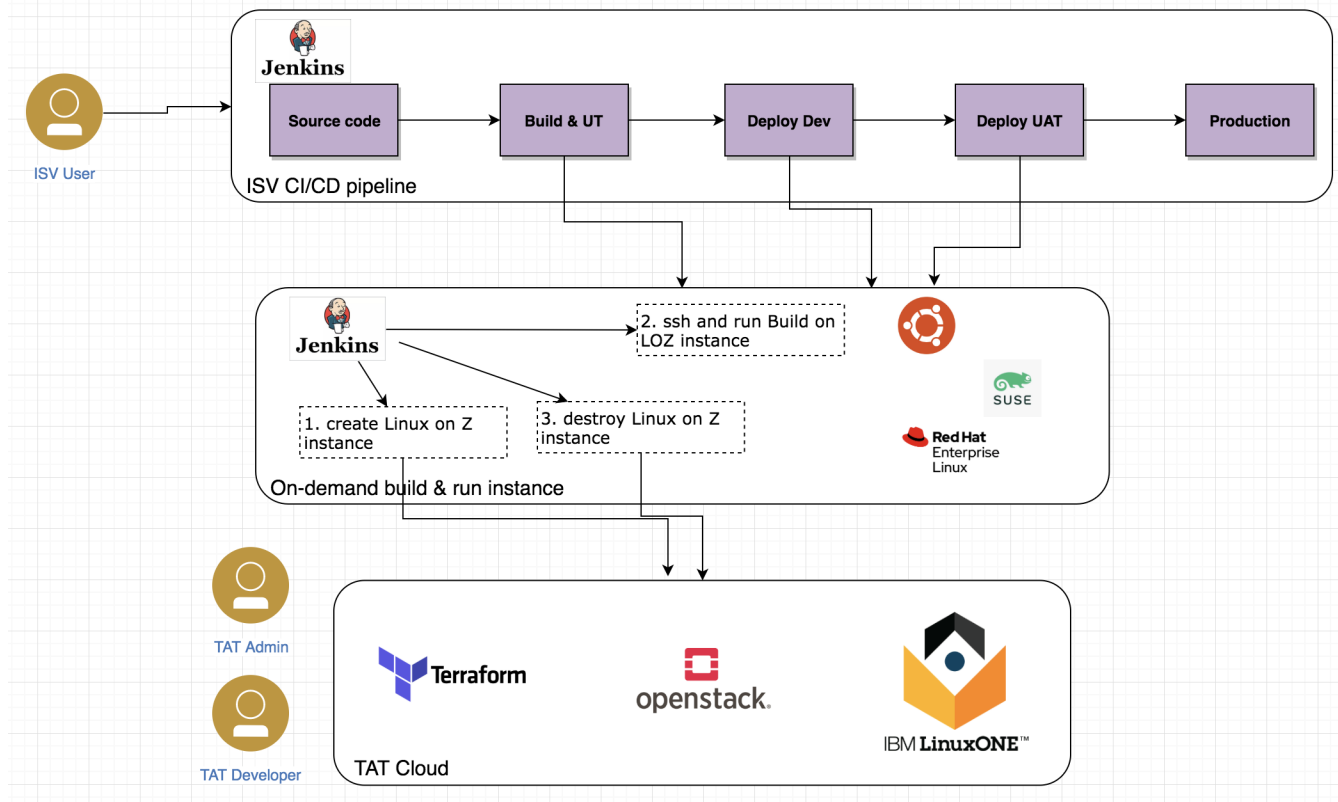
### 1. Ephemeral instances (aka dynamic or temporary instances)

- Ensures pipelines scalability, availability, reproducibility and also efficiency. Instances are created clean from the base image for each job and destroyed upon job completion.
- Can have a storage volume mounted for preserving data between the runs.

### 2. Static instances (aka permanent instances)

- Good for compatibility with manual steps such as QA or steps with long preparation times such as database instances with pre-seeded test data. Instances are created one time and then preserved on the disk between uses.

## Architecture



## Environment configuration

### Linux on Z instances provisioned on demand

- ☒ LoZ Instances with t-shirt size specs (to be specified in `terraform.tfvars`):
  - xsmall (1 vCPU, 1 GB RAM, 10 GB Disk)
  - small (2 vCPUs, 2 GB RAM, 20 GB Disk)
  - medium (2 vCPUs, 8 GB RAM, 40 GB Disk)
  - large (4 vCPUs, 16 GB RAM, 50 GB Disk)
  - xlarge (8 vCPUs, 16 GB RAM, 50 GB Disk)
  - Medium (2 vCPUs, 8GB RAM, 20GB Disk)
  - RHEL 8.2/8.3, Ubuntu 18.04/20.04, Fedora 32, RHEL CoreOS 4.x
  - ■ Other OS images can be prepared on request

### Prerequisites

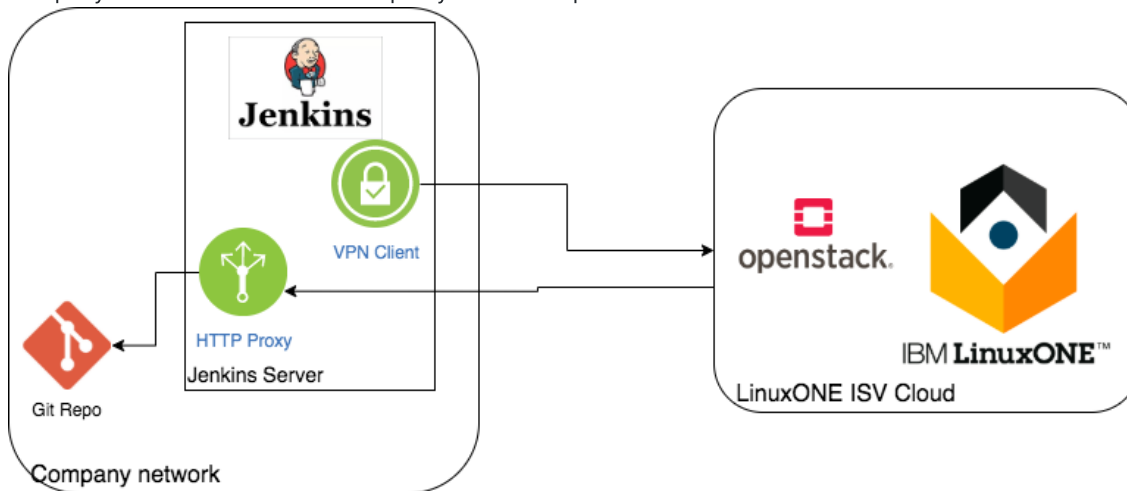
To run the Jenkins builds on a remote Linux on Z server, a few prerequisite programs must be installed on the Jenkins server host.

- ☐ Jenkins  $\geq$  v2.2
  - <http://mirrors.jenkins.io/war/latest/jenkins.war>
- ☐ OpenConnect version  $\geq$  v8.10
  - <http://www.infradead.org/openconnect/download.html>
- ☐ Terraform  $\geq$  v0.13.2
  - <https://www.terraform.io/downloads.html>
- ☐ Openstack  $\geq$  4.0.0

- <https://docs.openstack.org/newton/user-guide/common/cli-install-openstack-command-line-clients.html>
- `pip install python-openstackclient`

## Setup network between sites

Outbound connectivity to IBM LinuxONE ISV cloud is achieved via VPN point to site connection. Inbound connectivity to the company network is done via HTTP proxy that is setup on the Jenkins server where also VPN client runs.



### Network configuration setup

1. Connect to ISV VPN network to access Z/LinuxONE resources On the Jenkins server, run
  - `echo "VPN_key" | sudo openconnect --user=isv --passwd-on-stdin etpgw13.dfw.ibm.com`
2. [Optional] Proxy for company network access to access company resources, e.g. git repos You may want to setup a HTTP proxy on the Jenkins server, which also has the VPN client running. This would enable Jenkins pipeline jobs to access company resources such as git repos.
  - `git config --global http.proxy http://proxy.mycompany:8080`

## Installation and Linux Server Deployment

1. Get access to the LinuxOne ISV Cloud project
  - `http://172.29.191.130/`
2. Update Jenkinsfile with the assigned floating IP 172.29.128.x and ssh password
3. Install pre-requisites on Jenkins server.


#### 4. Create a new Pipeline item in Jenkins server

Dashboard > All >


### Enter an item name

loz-pi


loz-pipe

**Freestyle project**


This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this even used for something other than software build.

**Pipeline**


Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform specific builds, etc.

**Folder**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, creates a separate namespace, so you can have multiple things of the same name as long as they are in different folder

**GitHub Organization**

#### 5. Paste content of Jenkinsfile in Pipeline script field and modify

loz-pipe >

General Build Triggers **Advanced Project Options** Pipeline

Advanced...

### Pipeline

Definition Pipeline script

Script

```
4 node {
5     def remote = [:]
6     remote.name = 'test'
7     remote.host = '172.29.191.x' //IP assigned to the ISV
8     remote.user = 'fedora'
9     remote.password = 'REPLACE_ME'
10    remote.allowAnyHosts = true
11
12    stages {
13        stage('create-loz-instance') {
14            steps {
15                writeFile file: 'main.tf', text: ''
16            }
17        }
18        provider "openstack" {
19            auth_url = "http://172.29.191.130:5000/v3"
20            cloud = "iron-v3"
21        }
22    }
23 }
```

☒ Use Groovy Sandbox

[Pipeline Syntax](#)

#### 6. Run the pipeline

### Script to create and destroy Linux on Z instace

Instances are created for the single run of the pipeline or steps in the Ephemeral instances scenario.

1. Prepare main.tf, terraform.tfvars and clouds.yaml files in the current directory
2. Run terraform init and apply to provision Linux on Z instace

```
terraform init
terraform apply --auto-approve
```

3. Preserve the state file terraform.tfstate
4. SSH to the instance and run build/test shell scripts

```
ssh fedora@172.29.191.x
```

5. Run terraform destroy to deprovision Linux on Z instace

```
terraform destroy --auto-approve
```

## Script to shelve and unshelve Linux on Z instace

Instances are shelved between the pipeline runs in the Static instances scenario. Shelving is a feature of the OpenStack that is made available in LinuxONE ISV Cloud. Refer the official documentation here <https://docs.openstack.org/ocata/user-guide/cli-stop-and-start-an-instance.html>

1. List server instances using openstack cli command

```
#>openstack server list --os-cloud z-isv-project -c Name -c Status
```

Name	Status
master-2	ACTIVE
worker-0	ACTIVE
worker-1	ACTIVE
bootstrap	ACTIVE
master-0	ACTIVE
master-1	ACTIVE
bastion	ACTIVE

2. Shelf Linux on Z instace by the name

```
#>openstack server shelve bootstrap --os-cloud z-isv-project
```

```
#>openstack server list --os-cloud z-isv-project -c Name -c Status
```

Name	Status
master-2	ACTIVE
worker-0	ACTIVE
worker-1	ACTIVE
bootstrap	SHELVED_OFFLOADED
master-0	ACTIVE
master-1	ACTIVE
bastion	ACTIVE

3. Unshelve the instance before using it again

```
#>openstack server unshelve bootstrap --os-cloud z-isv-project
```

```
#>openstack server list --os-cloud z-isv-project -c Name -c Status
```

Name	Status
master-2	ACTIVE
worker-0	ACTIVE
worker-1	ACTIVE
bootstrap	ACTIVE
master-0	ACTIVE
master-1	ACTIVE

bastion	ACTIVE	
+-----+	+-----+	+