



Please

**Ask questions
through the app**



Rate Session

Thank you!



Building a Secure Future.™

Jenkins in the cloud

Self-healing, highly scalable and secure



Bhagyashri
Sarbhukan
DevOps
Engineer



Jan-Jaap
Oosterwijk
Technology
Evangelist

IRDETO IS THE **WORLD** LEADER IN DIGITAL PLATFORM SECURITY



NEARLY **50 YEARS** OF SECURITY
EXPERTISE IN PAY MEDIA



+5 BILLION DEVICES AND
APPLICATIONS SECURED



SERVING **400+ CUSTOMERS**
IN 75+ COUNTRIES



219 PATENTS & 285 PATENTS
PENDING



NEARLY 1,000 SECURITY
EXPERTS EMPLOYED



70% OF EMPLOYEES ARE IN
ENGINEERING/RESEARCH/
DEVELOPMENT



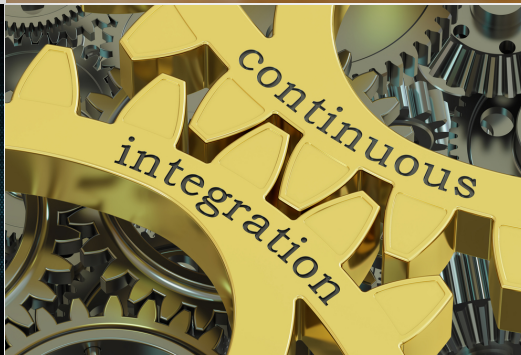
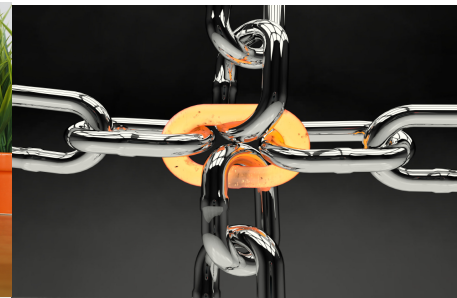
+15 LOCATIONS COVERING
6 CONTINENTS

IRDETO'S VISION



To build a secure future where people can embrace connectivity without fear.

Irdeto protects platforms and applications for media & entertainment, gaming, connected transport, connected spaces and IoT connected industries.



Agenda

- ✓ Jenkins recap
- ✓ Jenkins on-premise
- ✓ Jenkins in the cloud
- ✓ Coding Jenkins in AWS
- ✓ Way forward
- ✓ Demo
- ✓ Q & A

Jenkins recap



Why Jenkins?

A word cloud centered around the words 'RELEASE' and 'MANAGEMENT'. The words are arranged in a circular pattern around the central text. The words are in various sizes and orientations, with some being bold and red, and others in a smaller, grey font. The words include: DEVELOPMENT, DEPLOYMENT, TOOLS, SCHEDULING, TESTING, INTEGRATION, SOFTWARE, CONTINUOUS, AUTOMATION, PROJECT, CONTROLLING, and PLANNING. The words 'RELEASE' and 'MANAGEMENT' are the largest and most prominent, with 'RELEASE' in red and 'MANAGEMENT' in black. The word 'MANAGING' is also large and black, positioned below 'MANAGEMENT'. The word 'TESTING' is large and black, positioned to the left of 'MANAGEMENT'. The word 'DEVELOPMENT' is large and black, positioned above 'DEPLOYMENT'. The word 'DEPLOYMENT' is large and black, positioned above 'MANAGEMENT'. The word 'TOOLS' is large and black, positioned to the right of 'DEPLOYMENT'. The word 'SCHEDULING' is large and black, positioned to the right of 'TOOLS'. The word 'INTEGRATION' is large and black, positioned to the left of 'TESTING'. The word 'SOFTWARE' is large and black, positioned to the left of 'MANAGING'. The word 'CONTINUOUS' is large and black, positioned to the left of 'MANAGING'. The word 'AUTOMATION' is large and black, positioned below 'MANAGING'. The word 'PROJECT' is large and black, positioned to the right of 'MANAGING'. The word 'CONTROLLING' is large and black, positioned to the right of 'MANAGING'. The word 'PLANNING' is large and black, positioned to the right of 'MANAGING'.

DEVELOPMENT
DEPLOYMENT
TOOLS
SCHEDULING
RELEASE
INTEGRATION
TESTING **MANAGEMENT** **MANAGING** **PLANNING**
SOFTWARE
CONTINUOUS
AUTOMATION
PROJECT
CONTROLLING

Jenkins1.x/Jenkins 2.x

Node

Master

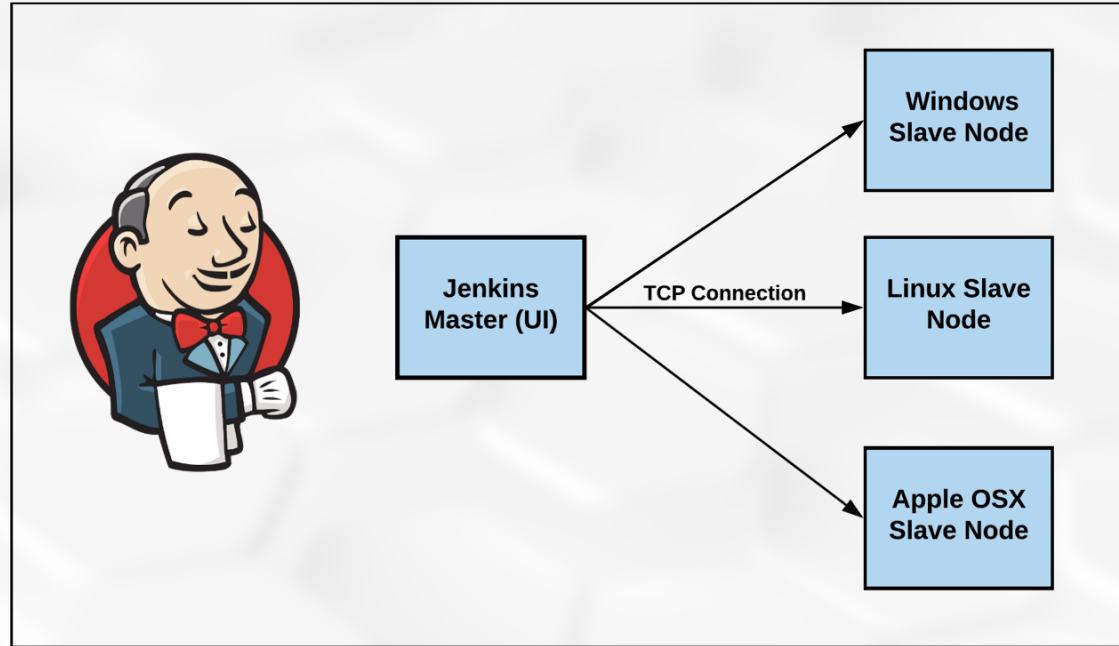
Slave

Agent

Executor

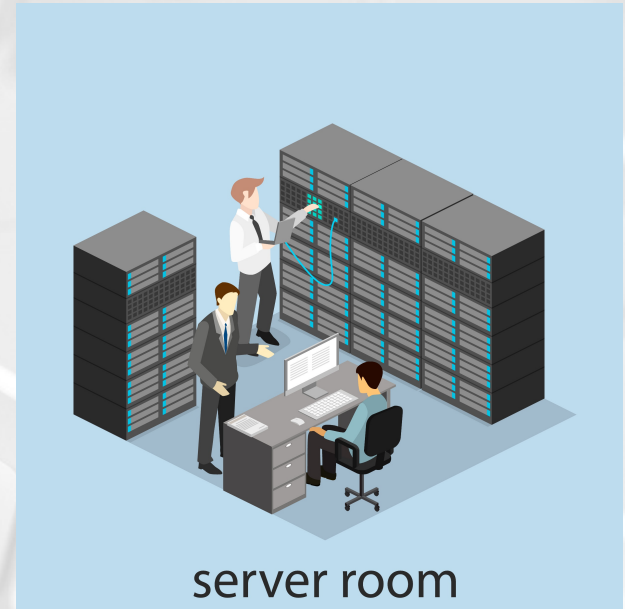


Jenkins Master/Slave architecture

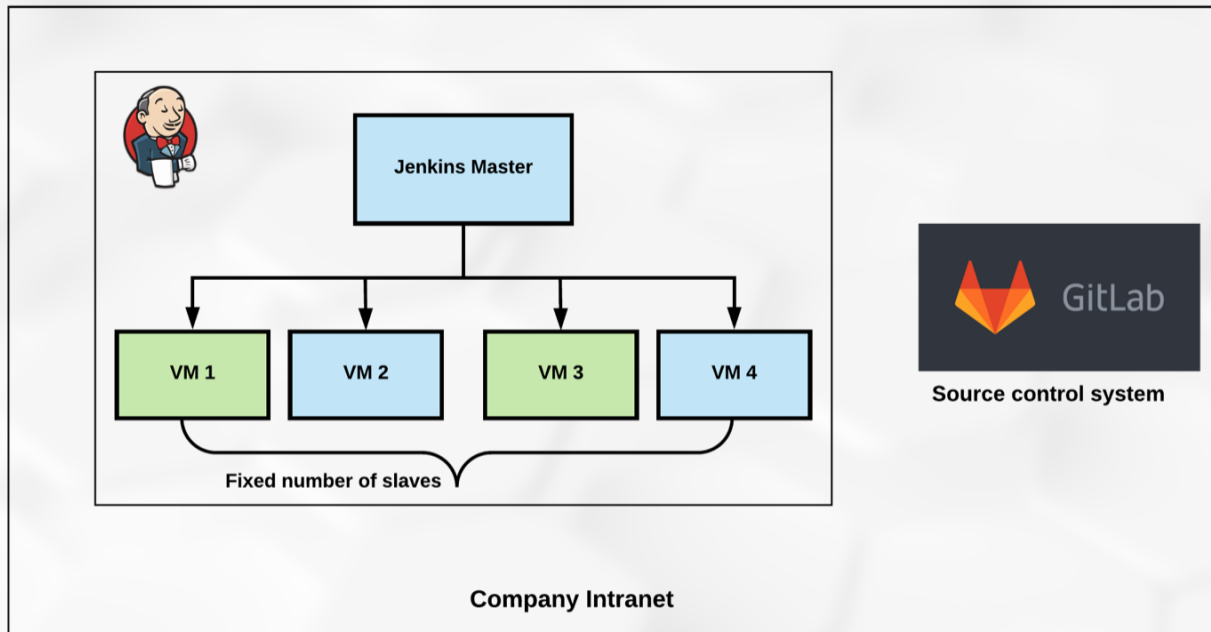


- ✓ Single Jenkins master
- ✓ Multiple slaves
- ✓ Master delegates jobs to an executor available in one of the slaves/agents
- ✓ Slaves/agents runs job
- ✓ Master is a single point of failure

Jenkins on-premise



Architecture



Risks with the on-premise Jenkins

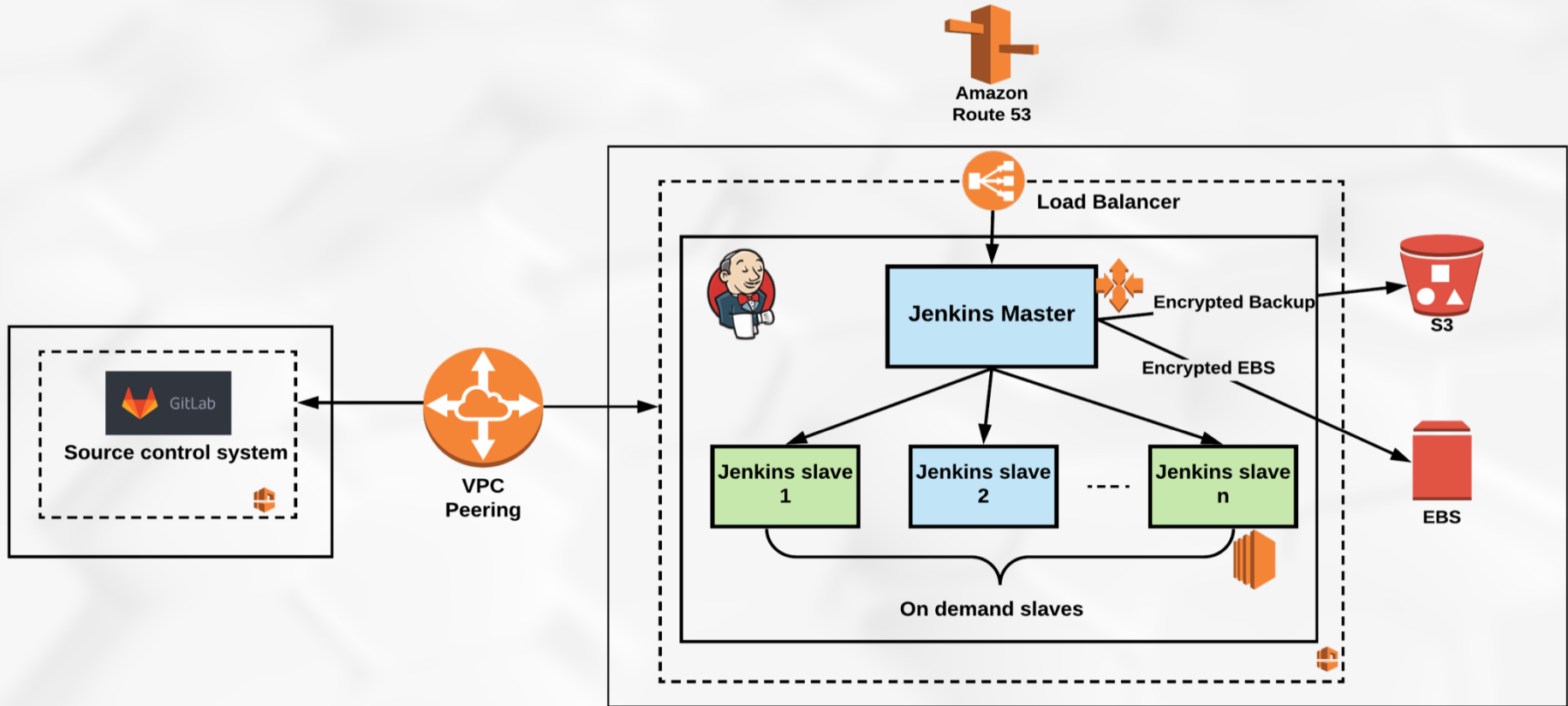
- ✓ Fixed number of slaves / agents – not ready for the spiky load
- ✓ Downtime can be in number of days in case of machine failure
- ✓ Manual setup
 - ✓ Undocumented knowledge
 - ✓ Cumbersome and prone to error
 - ✓ High maintenance effort – updates, replacing or fixing failing nodes, backup

Jenkins – in the cloud



Architecture

<https://goto-irdeto-jenkins.com>



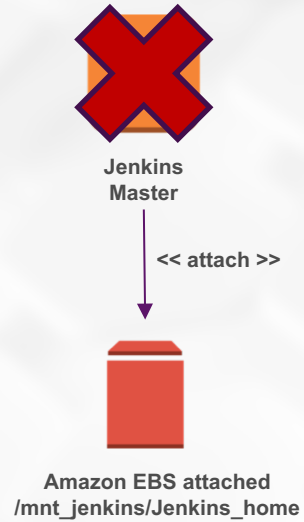
Advantages

Self-healing

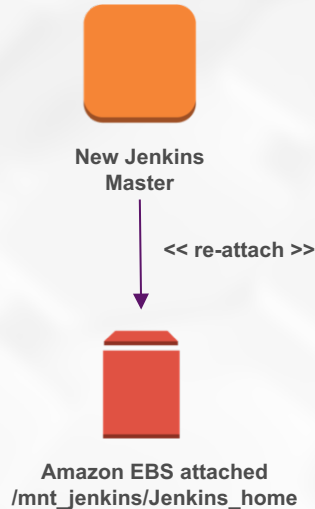
Highly Scalable

Secure

Self-healing

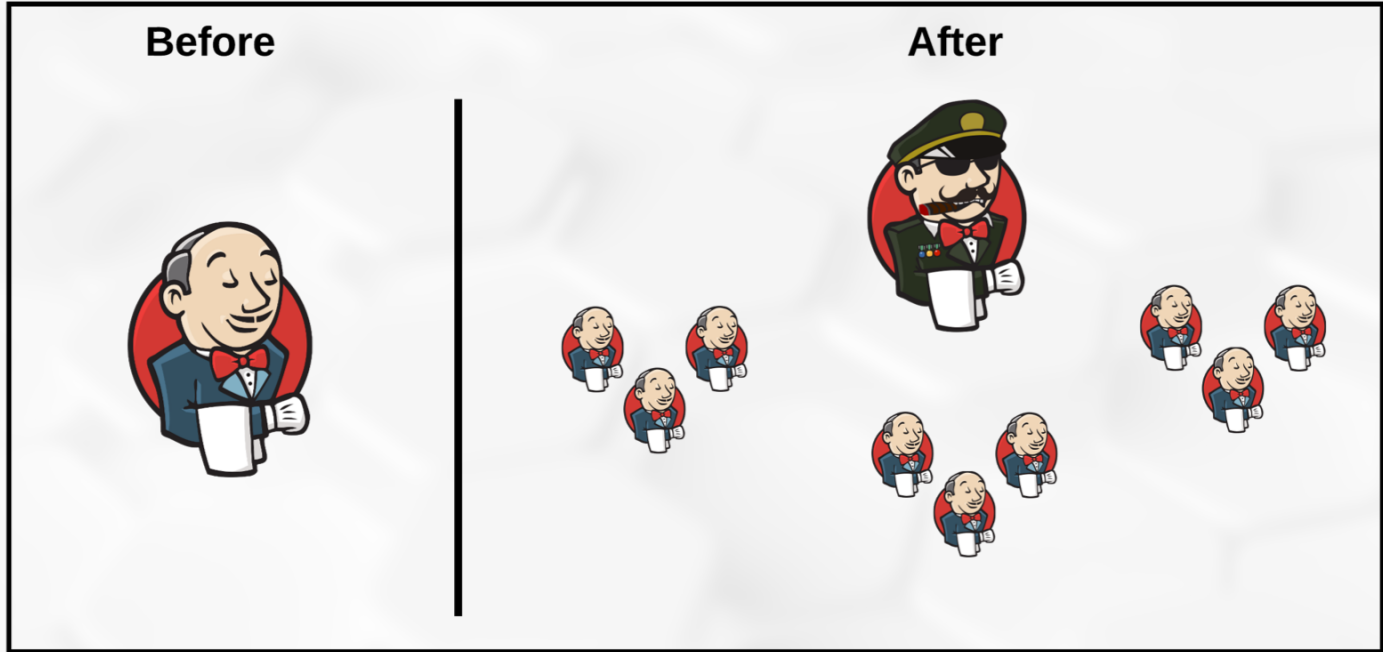


Self-healing



- ✓ Possible through Amazon Elastic Block Store (Amazon EBS), which provides persistent block storage volumes
- ✓ It is SSD or HDD backed
- ✓ Amazon backs it up and creates snapshots to S3
- ✓ EBS volume is designed for 99.999% availability

Highly Scalable



Highly Scalable

Description

AMI ID

Instance Type

EBS Optimized ☐

Availability Zone

☐ Use Spot Instance

Security group names

Remote FS root

Remote user

AMI Type

Root command prefix

Slave command prefix

Remote ssh port

Labels

Usage

Idle termination time

Init script

Number of Executors

JVM Options

Stop/Disconnect on Idle Timeout ☐

Subnet ID for VPC

Use dedicated tenancy ☐

Tags

EC2 Tag/Value Pairs

Use private DNS ☐

Instance Cap

IAM Instance Profile

Delete root device on instance termination ☐

Use ephemeral devices ☐

Block device mapping

Launch Timeout in seconds

Associate Public IP ☐

Connect using Public IP ☐

Connect by SSH Process ☐

Principle of least privilege

AWS IAM Role and Policies

Encrypted EBS and S3 backup

LDAP/AD integration



Coding in AWS

<https://github.com/bsarbhukan/self-healing-jenkins>



AWS CloudFormation

- AWS CloudFormation

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "This template deploys a ec2 instance to the provided VPC and subnet",
  "Parameters": {
    "DeploymentName": {
      "Description": "An environment name that will be prefixed to resource names",
      "Type": "String"
    },
    "KeyPair": {
      "Description": "Name of an existing EC2 KeyPair to enable SSH access to the instance",
      "Type": "AWS::EC2::KeyPair::KeyName",
      "ConstraintDescription": "Must be a valid EC2 keypair."
    },
    "SubnetId": {
      "Description": "Management Subnet for the instance",
      "Type": "AWS::EC2::Subnet::Id"
    },
    "VpcId": {
      "Description": "VPC ID for resources",
      "Type": "AWS::EC2::VPC::Id"
    },
    "DiskSize": {
      "Description": "Root disk size",
      "Type": "Number",
      "Default": 100
    },
    "InstanceType": {
      "Description": "EC2 instance type",
      "Type": "String",
      "Default": "t2.small",
      "AllowedValues": [...],
      "ConstraintDescription": "must be a valid EC2 instance type."
    },
    "SecurityGroups": {
      "Description": "Select the existing Security Groups to use for new host",
      "Type": "List<AWS::EC2::SecurityGroup::Id>"
    },
    "AMIID": {
      "Description": "AMI reference for ubuntu",
      "Type": "String",
      "Default": "ami-xxxxxx"
    }
  },
  "Resources": {
    "MyNewEC2Instance": {
      "Type": "AWS::EC2::Instance",
      "Properties": {
        "InstanceType": { "Ref": "InstanceType" },
        "KeyName": { "Ref": "KeyPair" },
        "ImageId": { "Ref": "AMIID" },
        "SubnetId": { "Ref": "SubnetId" },
        "SecurityGroupIds": {
          "Fn::Split": [ " ",
            {
              "Fn::Join": [ " ",
                [
                  { "Fn::Join": [ " ",
                    [
                      { "Fn::Join": [ " ",
                        [
                          { "Ref": "SecurityGroups" }
                        ]
                      ]
                    ]
                  ]
                ]
              ]
            }
          ]
        }
      ]
    },
    "Tags": [
      { "Key": "Application", "Value": { "Ref": "AWS::StackId" } },
      { "Key": "Name", "Value": { "Fn::Sub": "${DeploymentName}-instance-01" } }
    ],
    "BlockDeviceMappings": [
      {
        "DeviceName": "/dev/sda1",
        "Ebs": { "VolumeSize": { "Ref": "DiskSize" } }
      }
    ],
    "UserData": {
      "Fn::Base64": {
        "Fn::Join": [
          "",
          [
            "#!/bin/bash -v\n",
            "mkdir /tmp/automation\n"
          ]
        ]
      }
    }
  ]
}
```

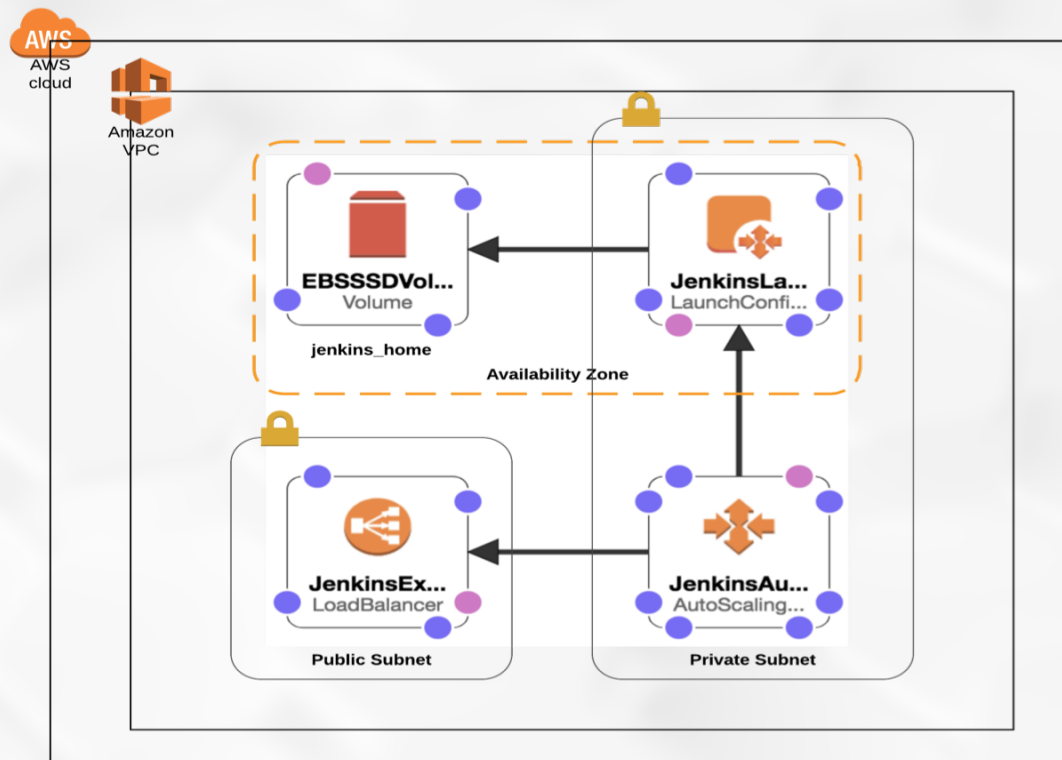
EC2 instance type

```
Resources: {
  "MyNewEC2Instance": {
    "Type": "AWS::EC2::Instance",
    "Properties": {
      "InstanceType": { "Ref": "InstanceType" },
      "KeyName": { "Ref": "KeyPair" },
      "ImageId": { "Ref": "AMIID" },
      "SubnetId": { "Ref": "SubnetId" },
      "SecurityGroupIds": {
        "Fn::Split": [ " ",
          [
            { "Fn::Join": [ " ",
              [
                { "Fn::Join": [ " ",
                  [
                    { "Ref": "SecurityGroups" }
                  ]
                ]
              ]
            ]
          ]
        ]
      ],
      "Tags": [
        { "Key": "Application", "Value": { "Ref": "AWS::StackId" } },
        { "Key": "Name", "Value": { "Fn::Sub": "${DeploymentName}-instance-01" } }
      ],
      "BlockDeviceMappings": [
        {
          "DeviceName": "/dev/sda1",
          "Ebs": { "VolumeSize": { "Ref": "DiskSize" } }
        }
      ],
      "UserData": {
        "Fn::Base64": {
          "Fn::Join": [
            "",
            [
              "#!/bin/bash -v\n",
              "mkdir /tmp/automation\n"
            ]
          ]
        }
      }
    }
  ]
}
```

EC2 instance creation

Refer Parameters

Jenkins master, EBS, ASG and ELB



Parameters

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "This creates launch configuration for jenkins master machine using user data",
  "Parameters": {
    "MyServiceVPC": {"Description": "VPC with some useful service tools"...},
    "KeyNameTest": {"Description": "Name of an existing EC2 KeyPair to enable SSH access to the"},
    "InstanceType": {"Description": "EC2 instance type"...},
    "AMI": {"Description": "AMI reference for ubuntu AMI id"...},
    "ELBSecurityGroup": {"Description": "Access to ELBs from port 443 vpn address range"...},
    "SSHSGFromJmpHost": {"Description": "Allow SSH connection from jumpbox server"...},
    "SubnetPri1b": {"Description": "Private subnet"...},
    "SubnetPub1b": {"Description": "Public subnet"...},
    "ENV": {"Description": "Environment name"...},
    "ACMIdentifier": {"Description": "Identifier of ACM certificate which will be used for LB"},
    "EBSVolumeID": {"Default": "create"...},
    "EBSVolumeSize": {"Default": "120"...},
    "AvailabilityZone": {"Description": "Availability Zone."...},
    "RegionPrefix": {"Description": "AWS region. e.g. dub for dublin, ore for oregon"...},
    "HostedZoneId": {"Description": "HostedZone for the Domain you would like to use"...},
    "DnsZone": {"Description": "Used DNSZone"...}
  },
}
```

Availability Zone

EBS volume

```
},
"EBSSSDVolume": {
  "Type": "AWS::EC2::Volume",
  "Properties": {
    "VolumeType": "gp2",
    "Encrypted": true,
    "Size": {"Ref": "EBSVolumeSize"},
    "AvailabilityZone": {"Ref": "AvailabilityZone"},
    "Tags": [{"Key": "Name", "Value": {"Fn::Join": ["", [{"Ref": "ENV"}, "-jenkins-volume"]]}]},
    "Condition": "CreateEBSVolume"
  }
},
"Conditions": {
  "CreateEBSVolume": {"Fn::Equals": [{"Ref": "EBSVolumeID"}, "create"]}
},
```

Encrypted EBS volume creation

EBS volume

```
},
"EBSSSDVolume": {
  "Type": "AWS::EC2::Volume",
  "Properties": {
    "VolumeType": "gp2",
    "Encrypted": true,
    "Size": {"Ref": "EBSVolumeSize"},
    "AvailabilityZone": {"Ref": "AvailabilityZone"},
    "Tags": [{"Key": "Name", "Value": {"Fn::Join": ["", [{"Ref": "ENV"}, "-jenkins-volume"]]}]},
    "Condition": "CreateEBSVolume"
  }
},
"Conditions": {
  "CreateEBSVolume": {"Fn::Equals": [{"Ref": "EBSVolumeID"}, "create"]}
```

Check for a condition to create
new volume or reuse existing EBS
volume

EBS volume

```
"EBS_VOLUME_ID=",
{
  "Fn::If": [
    "CreateEBSVolume",
    {
      "Ref": "EBSSSDVolume"
    },
    {
      "Ref": "EBSVolumeID"
    }
  ]
},
"\n",
"EBS_VOLUME_DEVICE=\"/dev/xvddh\"\n",
"MOUNT_POINT=\"/mnt_jenkins\"\n",
"INSTANCE_ID=$(ec2metadata --instance-id)\n",
"sudo apt-get update\n",
"sudo apt-get install -qy python-pip python-dev build-essential jq\n",
"sudo pip install --upgrade pip\n",
"sudo pip install --upgrade awscli\n",
"echo \"Attaching EBS volume ${EBS_VOLUME_ID} to device ${EBS_VOLUME_DEVICE} on instance\n",
"aws --region \"${Ref\": \"AWS::Region\"},\" ec2 attach-volume --volume-id ${EBS_VOLUME_ID}
```

Attach EBS volume to EC2 instance



EBS volume contd.

```

"echo \"Waiting for attach to complete...\n",
"let RETRIES_LEFT=60\n",
"while [[ \"$RETRIES_LEFT\" -gt \"0\" ]]; do\n",
"    sleep 1s\n",
"    ATTACH_STATUS=$(aws --region \"${Ref}: AWS::Region\" ec2 describe-volumes\n",
"    if [ \"${ATTACH_STATUS}\" == \"attached\" ]; then\n",
"        echo \"Attach completed.\n",
"        break\n",
"    fi\n",
"    echo \"Current status: ${ATTACH_STATUS}.\n",
"    let \"RETRIES_LEFT=RETRIES_LEFT-1\n",
"done\n",
"set +e\n",
"sudo file -s ${EBS_VOLUME_DEVICE} | cut -d , -f1 | grep -q \"ext4\"\n",
"if [ $? -eq 0 ]; then\n",
"    echo \"Already data in the EBS volume ...\"\n",
"else\n",
"    sudo mkfs -t ext4 ${EBS_VOLUME_DEVICE}\n",
"fi\n",
"set -e\n",
"sudo mkdir ${MOUNT_POINT}\n",
"sudo mount ${EBS_VOLUME_DEVICE} ${MOUNT_POINT}\n",

```

Wait for EBS to get attached

EBS volume

```

"echo \"Waiting for attach to complete...\"\\n",
"let RETRIES_LEFT=60\\n",
"while [[ \"$RETRIES_LEFT\" -gt \"0\" ]]; do\\n",
"    sleep 1s\\n",
"    ATTACH_STATUS=$(aws --region \"${Ref}: \"AWS::Region\", \" ec2 describe-volumes
"    if [ \"${ATTACH_STATUS}\" == \"attached\" ]; then\\n",
"        echo \"Attach completed.\"\\n",
"        break \\n",
"    fi\\n",
"    echo \"Current status: ${ATTACH_STATUS}.\"\\n",
"    let \"RETRIES_LEFT=RETRIES_LEFT-1\"\\n",
"done\\n",
"set +e\\n",
"sudo file -s ${EBS_VOLUME_DEVICE} | cut -d , -f1 | grep -q \"ext4\"\\n",
"if [ $? -eq 0 ]; then\\n",
"    echo \"Already data in the EBS volume ...\"\\n",
"else\\n",
"    sudo mkfs -t ext4 ${EBS_VOLUME_DEVICE}\\n",
"fi\\n",
"set -e\\n",
"sudo mkdir ${MOUNT_POINT}\\n",
"sudo mount ${EBS_VOLUME_DEVICE} ${MOUNT_POINT}\\n",

```

Check if FS already exists?

EBS volume

```

"echo \"Waiting for attach to complete...\"\\n",
"let RETRIES_LEFT=60\\n",
"while [[ \"$RETRIES_LEFT\" -gt \"0\" ]]; do\\n",
"    sleep 1s\\n",
"    ATTACH_STATUS=$(aws --region \"${Ref}: AWS::Region\" ec2 describe-volumes --volume-id \"${EBS_VOLUME_ID}\" --region \"${Region}\" | grep -q \"attached\" && echo \"attached\" || echo \"not attached\")\\n",
"    if [ \"$ATTACH_STATUS\" == \"attached\" ]; then\\n",
"        echo \"Attach completed.\"\\n",
"        break \\n",
"    fi\\n",
"    echo \"Current status: ${ATTACH_STATUS}.\"\\n",
"    let \"RETRIES_LEFT=RETRIES_LEFT-1\"\\n",
"done\\n",
"set +e\\n",
"sudo file -s ${EBS_VOLUME_DEVICE} | cut -d , -f1 | grep -q \"ext4\"\\n",
"if [ $? -eq 0 ]; then\\n",
"    echo \"Already data in the EBS volume ...\"\\n",
"else\\n",
"    sudo mkfs -t ext4 ${EBS_VOLUME_DEVICE}\\n",
"fi\\n",
"set -e\\n",
"sudo mkdir ${MOUNT_POINT}\\n",
"sudo mount ${EBS_VOLUME_DEVICE} ${MOUNT_POINT}\\n",

```

Else, create a new File system on newly created EBS

EBS volume

```

"echo \"Waiting for attach to complete...\"\\n",
"let RETRIES_LEFT=60\\n",
"while [[ \"$RETRIES_LEFT\" -gt \"0\" ]]; do\\n",
"    sleep 1s\\n",
"    ATTACH_STATUS=$(aws --region \"${Ref}: \"AWS::Region\"\", \"ec2 describe-volumes\"
"    if [ \"${ATTACH_STATUS}\" == \"attached\" ]; then\\n",
"        echo \"Attach completed.\"\\n",
"        break \\n",
"    fi\\n",
"    echo \"Current status: ${ATTACH_STATUS}.\"\\n",
"    let \"RETRIES_LEFT=RETRIES_LEFT-1\"\\n",
"done\\n",
"set +e\\n",
"sudo file -s ${EBS_VOLUME_DEVICE} | cut -d , -f1 | grep -q \"ext4\"\\n",
"if [ $? -eq 0 ]; then\\n",
"    echo \"Already data in the EBS volume ...\"\\n",
"else\\n",
"    sudo mkfs -t ext4 ${EBS_VOLUME_DEVICE}\\n",
"fi\\n",
"set -e\\n",
"sudo mkdir ${MOUNT_POINT}\\n",
"sudo mount ${EBS_VOLUME_DEVICE} ${MOUNT_POINT}\\n",

```

Mount EBS volume device to a mount point

Plugins installation


```
"# The plugins.txt file\n",  
"cat <<-EOF >/home/ubuntu/plugins.txt\n",  
"structs:1.10\n",  
"workflow-aggregator:2.5\n",  
"antisamy-markup-formatter:1.5\n",  
"maven-plugin:2.17\n",  
"handlebars:1.1.1\n",  
"workflow-cps-global-lib:2.9\n",  
"mapdb-api:1.0.9.0\n",  
"pipeline-milestone-step:1.3.1\n",  
"workflow-scm-step:2.6\n",  
"
```

Plugins are installed automatically during bootstrap

Click to add text

Jenkins master bootstrapping

```
"sudo apt-get update\n",  
"sudo apt-get install -qy python-pip apt-transport-https curl ca-certificates software-p  
"curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -\n",  
"sudo add-apt-repository \"deb [arch=amd64] https://download.docker.com/linux/ubuntu $(ls  
"sudo apt-get update\n",  
"sudo apt-get install -qy docker-ce\n",  
"sudo usermod -aG docker $USER\n",  
"\n",  
"sudo docker build -t jenkins_custom_made:v1 /home/ubuntu\n",  
"\n",  
"sudo mkdir -p ${MOUNT_POINT}/jenkins_home\n",  
"sudo chown ubuntu:ubuntu ${MOUNT_POINT}/jenkins_home\n",  
"sudo docker run --restart=always -d -p 8080:8080 -p 50000:50000 -v ${MOUNT_POINT}/jenkin  
"sudo add-apt-repository -y ppa:duplicity-team/ppa\n",  
"sudo apt-get update\n",  
"sudo apt-get --assume-yes install duplicity\n",  
"sudo apt-get --assume-yes install python-boto\n"
```

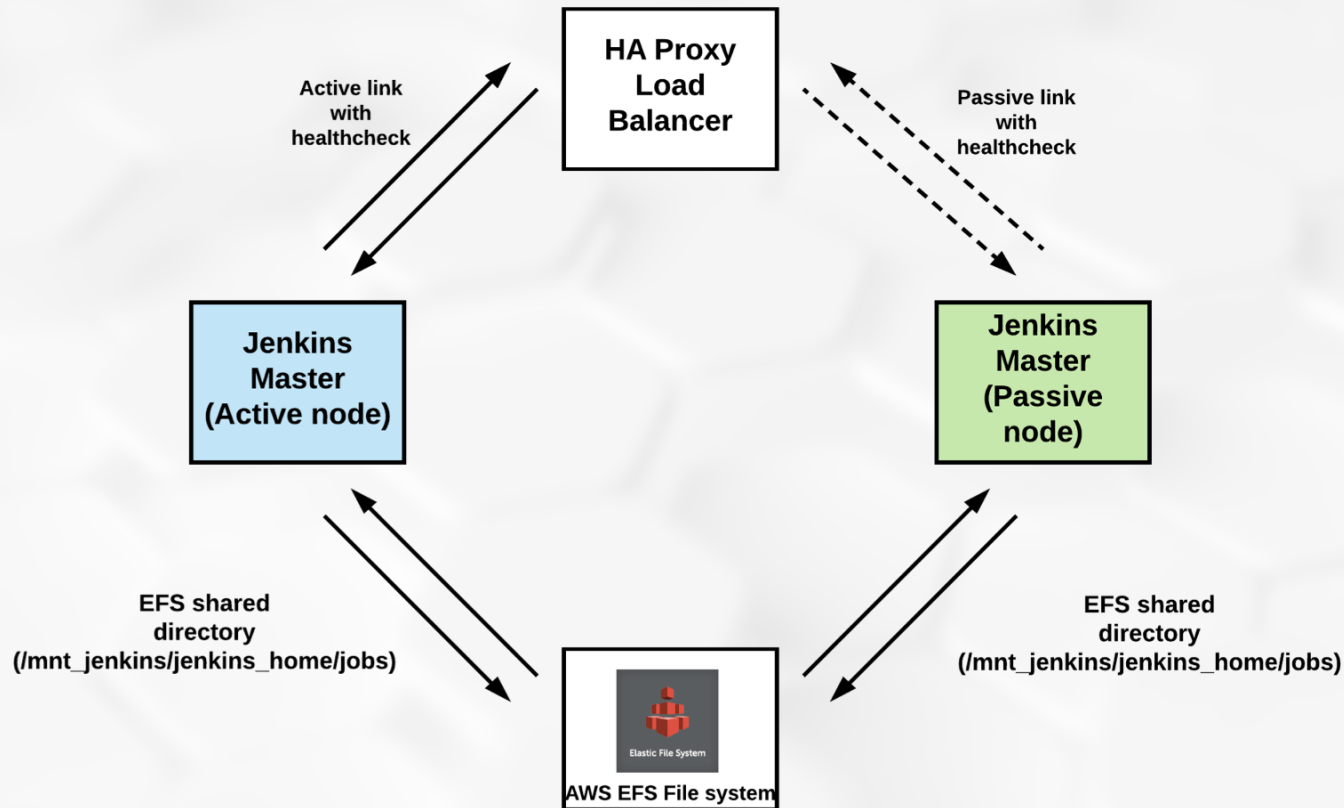


Build Jenkins docker image

Way forward...



High-availability



What can be improved?

- ✓ Automate high availability
- ✓ OS level LDAP integration
- ✓ S3 bucket deletion protection
- ✓ MFA protection for Jenkins master
- ✓ CloudWatch/Splunk integration with Jenkins jobs
- ✓ Instance level encryption using LUKS.
- ✓ IP table based firewall

Demo



Q & A





Building a Secure Future.™

THANK YOU!



please

**Remember to
rate this session**

Thank you!