SSM Document YAML (TKT 59) – Break Glass:

---

schemaVersion: "2.2"

description: "Command to create user using ssm document"

parameters:

UserName:

type: "String"

description: "Create user - break-glass"

default: "break-glass"

mainSteps:

- action: "aws:runShellScript"

name: "CreateUser"

inputs:

runCommand:

- sudo useradd break-glass

- cd /home/break-glass

- sudo mkdir .ssh

- sudo chmod 700 .ssh

- sudo echo -e "ssh-rsa  santiagorivera@MBP-M1.local" >> .ssh/authorized\_keys

- sudo chmod 600 .ssh/authorized\_keys

Splunk:

Warning: splunkforwarder-9.0.3-dd0128b1f8cd-linux-2.6-x86\_64.rpm: Header V4 RSA/SHA256 Signature, key ID b3cd4420: NOKEY

Complete

Checkmk – Ubuntu

Created new site ytmonitoring with version 2.1.0p20.cre.

The site can be started with omd start ytmonitoring.

The default web UI is available at http://ip-172-31-54-49/ytmonitoring/

The admin user for the web applications is cmkadmin with password: YuCwdpbM

For command line administration of the site, log in with 'omd su ytmonitoring'.

After logging in, you can change the password for cmkadmin with 'cmk-passwd cmkadmin'.

oot@ip-172-31-54-49:~# omd start pythonmonitoring

omd: The site 'pythonmonitoring' does not exist. You need to execute omd as root or site user.

root@ip-172-31-54-49:~# omd version

OMD - Open Monitoring Distribution Version 2.1.0p20.cre

root@ip-172-31-54-49:~# omd create pythonmonitoring

Adding /opt/omd/sites/pythonmonitoring/tmp to /etc/fstab.

Creating temporary filesystem /omd/sites/pythonmonitoring/tmp...OK

Updating core configuration...

Generating configuration for core (type nagios)...

Precompiling host checks...OK

Executing post-create script "01\_create-sample-config.py"...OK

Restarting Apache...OK

Created new site pythonmonitoring with version 2.1.0p20.cre.

The site can be started with omd start pythonmonitoring.

The default web UI is available at http://ip-172-31-54-49/pythonmonitoring/

The admin user for the web applications is cmkadmin with password: hiYpfCTp

For command line administration of the site, log in with 'omd su pythonmonitoring'.

After logging in, you can change the password for cmkadmin with 'cmk-passwd cmkadmin'.

RDP:

~~When prompted, connect to your instance using the following details:~~

~~Public DNS~~

~~ec2-18-207-112-79.compute-1.amazonaws.com~~

~~User name~~

~~Administrator~~

~~Password~~

~~p.mU!P4Q%QWjYvJR;)c&0B%\*KEOLL1?@~~

~~If you've joined your instance to a directory, you can use your directory credentials to connect to your instance.~~

When prompted, connect to your instance using the following details:

User name

 Administrator

Password

 p.mU!P4Q%QWjYvJR;)c&0B%\*KEOLL1?@

[Fleet Manager Remote Desktop](https://us-east-1.console.aws.amazon.com/systems-manager/managed-instances/rdp-connect?region=us-east-1&instances=i-09e29235cfe92ed9f)

If you've joined your instance to a directory, you can use your directory credentials to connect to your instance.

New 09Feb23:

root@ip-172-31-60-94:~# omd version

OMD - Open Monitoring Distribution Version 2.1.0p18.cre

root@ip-172-31-60-94:~# omd create ytmonitoring

Adding /opt/omd/sites/ytmonitoring/tmp to /etc/fstab.

Creating temporary filesystem /omd/sites/ytmonitoring/tmp...OK

Updating core configuration...

Generating configuration for core (type nagios)...

Precompiling host checks...OK

Executing post-create script "01\_create-sample-config.py"...OK

Restarting Apache...OK

Created new site ytmonitoring with version 2.1.0p18.cre.

The site can be started with omd start ytmonitoring.

The default web UI is available at http://ip-172-31-60-94/ytmonitoring/

The admin user for the web applications is cmkadmin with password: fHxsHkY9

For command line administration of the site, log in with 'omd su ytmonitoring'.

After logging in, you can change the password for cmkadmin with 'cmk-passwd cmkadmin'.

RDP:

Connection Type

Connect using RDP clientDownload a file to use with your RDP client and retrieve your password.

Connect using Fleet ManagerConnect to your instance using Fleet Manager Remote Desktop.

When prompted, connect to your instance using the following details:

User name

 Administrator

Password

 4d-?Ft@t2NZF\*ta3s9VHlP5Mt\*pMP20k

Brew installation on mac

/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"

Chatgpt:

AWS CloudFormation sample EC2, VPC, InternetGateway, Subnets, Routing table, ASG, Launch Template, Target Group, Security Group, User data install Apache, any region, option to select instance type, option to select region, default instance type t2.micro, default region us-east-1a.

TKT-37 Userdata:

#!/bin/bash

sudo yum -y update

sudo yum install -y httpd

sudo systemctl start httpd

sudo systemctl enable httpd

sudo yum install -y git

sudo yum install ruby wget -y

cd /home/ec2-user

sudo wget https://aws-codedeploy-us-east-1.s3.us-east-1.amazonaws.com/latest/install

sudo chmod +x ./install

sudo ./install auto

sudo git config --system credential.helper '!aws codecommit credential-helper $@'

sudo git config --system credential.UseHttpPath true

sudo git config --system credential.UseHttpsPath true

sudo git config --system user.name "Mathewos"

sudo git config --system user.email "matmar2@yahoo.com"

cd /home/ec2-user

sudo git clone -b main <https://git-codecommit.us-east-1.amazonaws.com/v1/repos/TKT-MAT-37-CodeCommit-Repo> /home/ec2-user/TKT-MAT-37-CodeCommit-Repo

vpn site-to-site connection between your own account and the organization account using openswam

<https://www.youtube.com/watch?v=7tTrN8WXMlg&ab_channel=DigitalCloudTraining>

**Lambda needs a role that will allow it to stop and start EC2 instances in your account. Make sure your Role does not allow Lambda to perform other activities.**

Created policy “Start-Stop-Instance-Policy-TKT-62”

Created role “Start-Stop-Instance-Role-TKT-62”

**Policy:**

{  
"Version": "2012-10-17",  
"Statement": [  
{  
"Sid": "VisualEditor0",  
"Effect": "Allow",  
"Action": [  
"ec2:StartInstances",  
"ec2:StopInstances",  
"logs:PutLogEvents"  
],  
"Resource": [  
"arn:aws:license-manager::202618001640:license-configuration:",  
"arn:aws:ec2::202618001640:instance/",  
"arn:aws:logs::202618001640:log-group::log-stream:"  
]  
},  
{  
"Sid": "VisualEditor1",  
"Effect": "Allow",  
"Action": [  
"logs:CreateLogStream",  
"logs:CreateLogGroup"  
],  
"Resource": "arn:aws:logs::202618001640:log-group:\*"  
}  
]  
}

**Role thrust relationship/entities:**

{  
"Version": "2012-10-17",  
"Statement": [  
{  
"Effect": "Allow",  
"Principal": {  
"Service": "[lambda.amazonaws.com](http://lambda.amazonaws.com)"  
},  
"Action": "sts:AssumeRole"  
}  
]  
}

MAT-59 Requirements:

* Create an SSM document to deploy this user so that it can be used again in the future. In your SSM document include some commands that verify that the user has been added.
* Since the user will not be able to login with a password you will need to add the following public Key to the authorized key file.
* ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDg4Tss2eHWT2Z/6SYYMNb2hzmcl0hoxckGhTf3KVIVpRFrfRks3Qnh2oGycey2icwYfrun1vWm6rmUug9hZTVTHfsabj++xylEu/8XuQto0HeWp9tk4iwNku6EZ+JBzuR7iHrVZcwqSCQn5ln9SaKALO
* uCqoDNQq/gWIYPmZVE0WD+66pIfKuUvaADNpVecn4B+AGsfBgAeKVz63zXqxbR8Y3Hjl2rvVfGBSGj9kxykH9klI0ew3falGh3D5JNpxXRyWg2u9LkLXEqqXRJXF9JqRKT0ZqTLaprWqOe6U7DoHV8ktyYyE6F1WqwjU4g0f8+gYQ8DgQK8ijcwnZmnfAo
* 21RniftymQRoGx6rr3o0TshA61WwC/lI90jnLakt89KWlEvMtb4jWkL0a9NuaUQf21c29lYvQfLSRyqqMXanGrkcrhCG0iY39gwAYVPMotulycd4BEOxv9sVDCoeHcDkJZw4XYVpSJvI1+vh41gquJJLEuA3Sq997uxSrOSjCPncOqSeOd63TJYXYk1hR6
* rbkipoqwglDrRbWKFZo+aWYpBISDV/ap0a9S4c3ltRrwJN6BgI5ZvHWEw0wUQA/VKnRlFtnvNYylIlMFnDjzDhgRxpunvmPpaz1cl1uiM398mNuOyQikxxqZovdkEYveJWssOM/V5ulY9HYGTaRAEO1w== santiagorivera@MBP-M1.local

Here is a link with information on how to accomplish this:

<https://aws.amazon.com/premiumsupport/knowledge-center/new-user-accounts-linux-instance/>

SSM Document:

---

schemaVersion: "2.2"

description: "Command to create user using ssm document"

parameters:

UserName:

type: "String"

description: "Create user - break-glass"

default: "break-glass"

mainSteps:

- action: "aws:runShellScript"

name: "CreateUser"

inputs:

runCommand:

- sudo yum update -y

- sudo useradd break-glass

- sudo su - break-glass

- cd /home/break-glass

- mkdir .ssh

- chmod 700 .ssh

- echo -e "ssh-rsa  santiagorivera@MBP-M1.local" >> .ssh/authorized\_keys

- chmod 600 .ssh/authorized\_keys

# How to install Splunk on a Linux Instance in AWS

<https://www.youtube.com/watch?v=0CHaDfNI4Sg&ab_channel=EmekaakaOn1Productions>

Pritunl Client:

<https://client.pritunl.com/#install>

# AWS EC2 + Autoscaling + Load Balancer + CodeDeploy | Deploy Code At Scale | DevOps With AWS

<https://www.youtube.com/watch?v=Ekgi2HfnJcw&t=1636s&ab_channel=SandipDas>

Github:

HTTPS:

<https://github.com/matmar2/python.git>

Github CLI:

gh repo clone matmar2/python

ChatGPT

<https://openai.com/blog/chatgpt/>

# Setup an AWS Site-to-Site Virtual Private Network (VPN)

<https://www.youtube.com/watch?v=7tTrN8WXMlg>

# AWS VPC Peering Connection Concept with Demo | VPC Peering | AWS VPC Peering Step by Step | AWS Demo

<https://www.youtube.com/watch?v=q-NTKPb16SM>

Terraform for beginners:

<https://kodekloud.com/courses/lab-terraform-for-beginners/>

# ChatGPT Tutorial - Use ChatGPT for DevOps tasks to 10x Your Productivity

<https://www.youtube.com/watch?v=l-kE11fhfaQ>

Checkmk:

<https://checkmk.com/product/features>

# Episode 1: Installing Checkmk and monitoring your first host

<https://www.youtube.com/watch?v=opO-SOgOJ1I&ab_channel=Checkmk>

# How to monitor your Linux servers with Checkmk

<https://www.techrepublic.com/article/how-to-monitor-your-linux-servers-with-checkmk/>

Apprenticeship Site – Procore

<https://sites.google.com/procoreplus.com/apprenticeship-site-aws/tool-guide?authuser=0>

# [ AWS 14 ] Set up your first AWS CodeCommit Repository

<https://www.youtube.com/watch?v=E1GJqfIEJkM&ab_channel=JustmeandOpensource>

### openSUSE, SUSE

<https://packages.cisofy.com/community/#centos-rhel>

# (SAA-C02) AWS Certified Solutions Architect - Associate

<https://www.youtube.com/watch?v=6S0_rNYzcPs&ab_channel=ChienDuong>

Cloudformation and Terraform templates from existing AWS setup - Former2

[www.former2.com](http://www.former2.com)

# Terraform S3 Backend Best Practices

<https://technology.doximity.com/articles/terraform-s3-backend-best-practices>

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 4.16"

}

}

}

# Configure provider

provider "aws" {

region = "us-east-1"

}

# Create VPC

resource "aws\_vpc" "vpc" {

cidr\_block = "10.0.0.0/16"

instance\_tenancy = "default"

tags = {

Name = "prod-VPC"

}

}

# Create internet gateway

resource "aws\_internet\_gateway" "ig" {

vpc\_id = aws\_vpc.vpc.id

tags = {

Name = "ig-project"

}

}

# Create public subnets

resource "aws\_subnet" "public\_1" {

vpc\_id = aws\_vpc.vpc.id

cidr\_block = "10.0.1.0/24"

availability\_zone = "us-east-1a"

map\_public\_ip\_on\_launch = true

tags = {

Name = "public-1"

}

}

# Create route table to internet gateway

resource "aws\_route\_table" "project\_rt" {

vpc\_id = aws\_vpc.vpc.id

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.ig.id

}

tags = {

Name = "project-rt"

}

}

# Associate public subnets with route table

resource "aws\_route\_table\_association" "public\_route\_1" {

subnet\_id = aws\_subnet.public\_1.id

route\_table\_id = aws\_route\_table.project\_rt.id

}

# Create security groups

resource "aws\_security\_group" "public\_sg" {

name = "public-sg"

description = "Allow web and ssh traffic"

vpc\_id = aws\_vpc.vpc.id

ingress {

from\_port = 80

to\_port = 80

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

ingress {

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

}

# Security group for ALB

resource "aws\_security\_group" "alb\_sg" {

name = "alb-sg"

description = "security group for alb"

vpc\_id = aws\_vpc.vpc.id

ingress {

from\_port = "0"

to\_port = "0"

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

egress {

from\_port = "0"

to\_port = "0"

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

}

# Create ALB

resource "aws\_lb" "project\_alb" {

name = "alb"

internal = false

load\_balancer\_type = "application"

security\_groups = [aws\_security\_group.alb\_sg.id]

subnets = [aws\_subnet.public\_1.id, aws\_subnet.public\_2.id]

}

# Create ALB target group

resource "aws\_lb\_target\_group" "project\_tg" {

name = "project-tg"

port = 80

protocol = "HTTP"

vpc\_id = aws\_vpc.vpc.id

depends\_on = [aws\_vpc.vpc]

}

# Create target attachments

resource "aws\_lb\_target\_group\_attachment" "tg\_attach1" {

target\_group\_arn = aws\_lb\_target\_group.project\_tg.arn

target\_id = aws\_instance.web1.id

port = 80

depends\_on = [aws\_instance.web1]

}

# Create listener

resource "aws\_lb\_listener" "listener\_lb" {

load\_balancer\_arn = aws\_lb.project\_alb.arn

port = "80"

protocol = "HTTP"

default\_action {

type = "forward"

target\_group\_arn = aws\_lb\_target\_group.project\_tg.arn

}

}

// -----------------------------------------------

// Change USERDATA varible value after grabbing RDS endpoint info

// -----------------------------------------------

data "template\_file" "user\_data" {

template = file("userdata.sh")

vars = {

db\_username = var.database\_user

db\_user\_password = var.database\_password

db\_name = var.database\_name

db\_RDS = aws\_db\_instance.wordpressdb.endpoint

}

}

# Create ec2 instances

resource "aws\_instance" "web1" {

ami = "ami-0cff7528ff583bf9a"

instance\_type = "t2.micro"

key\_name = "Mat-internship-ticket-key"

availability\_zone = "us-east-1a"

vpc\_security\_group\_ids = [aws\_security\_group.public\_sg.id]

subnet\_id = aws\_subnet.public\_1.id

associate\_public\_ip\_address = true

user\_data = <<-EOF

#!/bin/bash

# variable will be populated by terraform template

db\_username=${db\_username}

db\_user\_password=${db\_user\_password}

db\_name=${db\_name}

db\_RDS=${db\_RDS}

# install LAMP Server

yum update -y

#install apache server and mysql client

yum install -y httpd

yum install -y mysql

#first enable php7.xx from amazon-linux-extra and install it

amazon-linux-extras enable php7.4

yum clean metadata

yum install -y php php-{pear,cgi,common,curl,mbstring,gd,mysqlnd,gettext,bcmath,json,xml,fpm,intl,zip,imap,devel}

#install imagick extension

yum -y install gcc ImageMagick ImageMagick-devel ImageMagick-perl

pecl install imagick

chmod 755 /usr/lib64/php/modules/imagick.so

cat <<EOF >>/etc/php.d/20-imagick.ini

extension=imagick

EOF

systemctl restart php-fpm.service

systemctl start httpd

user\_data = data.template\_file.user\_data.rendered

# Change OWNER and permission of directory /var/www

usermod -a -G apache ec2-user

chown -R ec2-user:apache /var/www

find /var/www -type d -exec chmod 2775 {} \;

find /var/www -type f -exec chmod 0664 {} \;

sudo yum -y install mariadb-server

sudo service mariadb start

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Installing Wordpress using WP CLI\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

curl -O https://raw.githubusercontent.com/wp-cli/builds/gh-pages/phar/wp-cli.phar

chmod +x wp-cli.phar

mv wp-cli.phar /usr/local/bin/wp

wp core download --path=/var/www/html --allow-root

wp config create --dbname=$db\_name --dbuser=$db\_username --dbpass=$db\_user\_password --dbhost=$db\_RDS --path=/var/www/html --allow-root --extra-php <<PHP

define( 'FS\_METHOD', 'direct' );

define('WP\_MEMORY\_LIMIT', '128M');

PHP

# Change permission of /var/www/html/

chown -R ec2-user:apache /var/www/html

chmod -R 774 /var/www/html

# enable .htaccess files in Apache config using sed command

sed -i '/<Directory "\/var\/www\/html">/,/<\/Directory>/ s/AllowOverride None/AllowOverride all/' /etc/httpd/conf/httpd.conf

#Make apache autostart and restart apache

systemctl enable httpd.service

systemctl restart httpd.service

echo WordPress Installed

tags = {

Name = "wordpress\_instance"

}

}

# Database subnet group

resource "aws\_db\_subnet\_group" "db\_subnet" {

name = "db-subnet"

subnet\_ids = [aws\_subnet.private\_1.id, aws\_subnet.private\_2.id]

}

# Create database instance

resource "aws\_db\_instance" "project\_db" {

allocated\_storage = 5

engine = "mysql"

engine\_version = "5.7"

instance\_class = "db.t3.micro"

identifier = "db-instance"

db\_name = "project\_db"

username = "admin"

password = "password"

db\_subnet\_group\_name = aws\_db\_subnet\_group.db\_subnet.id

vpc\_security\_group\_ids = [aws\_security\_rds.id]

publicly\_accessible = false

skip\_final\_snapshot = true

}

resource "aws\_security\_group" "rds" {

name = "rds"

description = "Allow RDS traffic from EC2 instances"

ingress {

from\_port = 3306

to\_port = 3306

protocol = "tcp"

security\_groups = [aws\_security\_group.public\_sg.id]

}

}

# Create an S3 bucket for the Terraform state files

resource "aws\_s3\_bucket" "terraform\_state\_bucket" {

bucket = "mat-78-terraform-bucket"

acl = "private"

versioning {

enabled = true

}

}

# Configure the Terraform backend to use S3

terraform {

backend "s3" {

bucket = "mat-78-terraform-bucket"

key = "terraform.tfstate"

region = "us-east-1"

}

}

# Outputs

# Ec2 instance public ipv4 address

output "ec2\_public\_ip" {

value = aws\_instance.web1.public\_ip

}

# Db instance address

output "db\_instance\_address" {

value = aws\_db\_instance.project\_db.address

}

# Getting the DNS of load balancer

output "lb\_dns\_name" {

description = "The DNS name of the load balancer"

value = "${aws\_lb.project\_alb.dns\_name}"

}

MAT-49

#!/bin/bash

# Update the package index

sudo yum update -y

# Install Apache Web Server

sudo yum install -y httpd

# Start the Apache Web Server

sudo service httpd start

# Enable the Apache Web Server to start at boot

sudo chkconfig httpd on

# Display a message

echo "\* \* \* \* \* \* \* W A R N I N G \* \* \* \* \* \* \* \* \* \*  
This computer system is the property of ProCore Plus. It is for authorized use only. By using this system, all users acknowledge notice of, and agree to comply with, the Acceptable Use of Information Technology Resources Policy (AUP).   Unauthorized or improper use of this system may result in administrative disciplinary action, civil charges/criminal penalties, and/or other sanctions as set forth in the AUP. By continuing to use this system you indicate your awareness of and consent to these terms and conditions of use. **LOG OFF IMMEDIATELY** if you do not agree to the conditions stated in this warning.

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*" > /var/www/html/index.html

sudo mkdir /mnt/efs

sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-0d6526450707e8392.efs.us-east-1.amazonaws.com:/ /mnt/efs

sudo useradd -c "Mathewos G" mat

#sudo passwd mat (abcd1234)

sudo mkdir -p /mnt/efs/mat

sudo chown mat:mat /mnt/efs/mat

sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-0d6526450707e8392.efs.us-east-1.amazonaws.com:/mat /home/mat

sudo mkdir /mnt/efs/mat/home-directories

# MAT-60: create an Ansible playbook to install the CloudWatch agent on all bastion hosts. The playbook should also put custom metric such as memory and disk space from the bastion host

---

- name: Install CloudWatch agent and send custom metrics on bastion hosts

hosts: local

gather\_facts: false

tasks:

- name: Gather a list of instances with the tag "EC2-TKT-49-54-Bastion Instance"

ec2\_instance\_info:

filters:

"tag:Name": "EC2-TKT-49-54-Bastion Instance"

register: bastion\_instances

- name: Install dependencies

package:

name: awslogs

state: present

delegate\_to: "{{ item.private\_ip }}"

loop: "{{ bastion\_instances.instances }}"

- name: Download the CloudWatch agent configuration file

command: aws s3 cp s3://my-cloudwatch-bucket/cloudwatch-agent-config.json /tmp/cloudwatch-agent-config.json

delegate\_to: "{{ item.private\_ip }}"

loop: "{{ bastion\_instances.instances }}"

- name: Install the CloudWatch agent

command: /usr/local/bin/aws cloudwatch configure-agent --input-path /tmp/cloudwatch-agent-config.json

delegate\_to: "{{ item.private\_ip }}"

loop: "{{ bastion\_instances.instances }}"

- name: Start the CloudWatch agent

service:

name: awslogs

state: started

enabled: yes

delegate\_to: "{{ item.private\_ip }}"

loop: "{{ bastion\_instances.instances }}"

- name: Send custom metrics - Memory Usage

shell: echo "{{ item.private\_ip }} memory\_usage `free -m | grep Mem | awk '{print $3/$2 \* 100.0}'`" | /usr/local/bin/aws cloudwatch put-metric-data --metric-name MemoryUsage --unit Percent --value `free -m | grep Mem | awk '{print $3/$2 \* 100.0}'` --dimensions InstanceId={{ item.instance\_id }}

delegate\_to: "{{ item.private\_ip }}"

loop: "{{ bastion\_instances.instances }}"

- name: Send custom metrics - Disk Space Utilization

shell: echo "{{ item.private\_ip }} disk\_utilization `df -h / | grep / | awk '{print $5}' | sed 's/%//g'`" | /usr/local/bin/aws cloudwatch put-metric-data --metric-name DiskSpaceUtilization --unit Percent --value `df -h / | grep / | awk '{print $5}' | sed 's/%//g'` --dimensions InstanceId={{ item.instance\_id }}

delegate\_to: "{{ item.private\_ip }}"

loop: "{{ bastion\_instances.instances }}"

Boto3 documentation – Available Services

<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/index.html>

SSH from Bastion server to Private instance:

ssh-add -K privatekey.pem    (-K for Mac -> agent forwarding -> keypair to be available for the private instance)  
ssh -A ec2-user@public-ipin Bastion host  
ssh ec2-user@private-ip

ssh-add -c privatekey.pem (for Windows)

ssh -A userid@remoteip -> -A used to forward the ssh-agent to the remote host

# Full Stack DevOps Environment Setup Bash Script for Ubuntu, CentOS and MacOS

<https://awstip.com/full-stack-devops-environment-setup-bash-script-for-ubuntu-and-macos-44cd4b68a34>

# ****Installation of LAMP stack on Amazon Linux OS on AWS Ec2 Instance****

<https://awstip.com/installation-of-lamp-stack-on-amazon-linux-os-on-aws-ec2-instance-741114892bea>

Copy from EC2 to S3 bucket – “s3-ticket-mat-59” (then download from S3)

sudo aws s3 cp main.tf s3://s3-ticket-mat-59/main-18Feb23.tf

Copy a directory from local directory to EC2 instance

sudo scp -r -i Mat-internship-ticket-key.pem /Users/matmar2/Downloads/ariclaw-master/ [ec2-user@18.204.196.117:/home/ec2-user](mailto:ec2-user@18.204.196.117:/home/ec2-user)

Copy from /home/ec2-user to local drive (current directory in the local drive).

sudo scp -i Mat-internship-ticket-key.pem ec2-user@54.175.5.201:/home/ec2-user/lynis\_audit\_2023-02-20\_16:45:06.log ./

Recall from history – list (say line number 85:

!85 :p

# Building Web Apps With Python Has Never Been Easier — Get Started With Pynecone

<https://medium.com/codingthesmartway-com-blog/building-web-apps-with-python-has-never-been-easier-get-started-with-pynecone-a9f60c1532c>

# Terraform Deploy a Two Tier Architecture in AWS

<https://levelup.gitconnected.com/terraform-two-tier-architecture-66fa07c8e325>

CDK example

<https://catalog.us-east-1.prod.workshops.aws/workshops/071bbc60-6c1f-47b6-8c66-e84f5dc96b3f/en-US>