

# Deployment 2

Kenneth Tan

## Jenkins on EC2

To begin the deployment, I had to start Jenkins on my EC2. I did this by accessing cmd on my Windows machine. I changed directory to .ssh, where the keys for accessing my EC2 instance lived.

```
C:\Users\Kenneth>cd .ssh

C:\Users\Kenneth\.ssh>dir
Volume in drive C has no label.
Volume Serial Number is E2D9-E9DE

Directory of C:\Users\Kenneth\.ssh

08/28/2021  04:37 PM    <DIR>          .
08/28/2021  04:37 PM    <DIR>          ..
08/28/2021  02:23 PM                1,704 ec2-jenkins-4.pem
08/28/2021  04:33 PM                1,700 ec2-jenkins-ubuntu-1.pem
08/31/2021  09:04 AM                1,466 known_hosts
               3 File(s)                4,870 bytes
               2 Dir(s) 229,073,399,808 bytes free
```

This will allow me to run

```
ssh -i "ec2-jenkins-4.pem" ec2-user@ec2-54-159-201-70.compute-1.amazonaws.com
```

and access my EC2 from cmd. I then run

```
sudo systemctl start jenkins
```

to start running jenkins on the instance and I use

```
systemctl status jenkins
```

to make sure it is running.

```
[ec2-user@ip-172-31-89-52 ~]$ sudo systemctl start jenkins
[ec2-user@ip-172-31-89-52 ~]$ systemctl status jenkins
● jenkins.service - LSB: Jenkins Automation Server
   Loaded: loaded (/etc/rc.d/init.d/jenkins; bad; vendor preset: disabled)
   Active: active (running) since Sat 2021-09-04 15:04:13 UTC; 24h ago
     Docs: man:systemd-sysv-generator(8)
  Process: 14632 ExecStop=/etc/rc.d/init.d/jenkins stop (code=exited, status=0/SUCCESS)
  Process: 20714 ExecStart=/etc/rc.d/init.d/jenkins start (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/jenkins.service
           └─20718 /etc/alternatives/java -Djava.awt.headless=true -DJENKINS_HOME=/var/lib/jenkins -jar /usr/lib/jenk...

Sep 04 15:04:13 ip-172-31-89-52.ec2.internal systemd[1]: Starting LSB: Jenkins Automation Server...
Sep 04 15:04:13 ip-172-31-89-52.ec2.internal jenkins[20714]: Starting Jenkins [ OK ]
Sep 04 15:04:13 ip-172-31-89-52.ec2.internal systemd[1]: Started LSB: Jenkins Automation Server.
```

If the status is showing running, then I can open the Jenkins web interface with the EC2 public IPv4 address and port number

## Jenkins Pipeline

After accessing Jenkins, I clicked on new item and I selected Pipeline to create a new Pipeline. I had named mine test-cron-job2. I selected “Build Periodically” in the Build Trigger section and I included this cron scheduling formula

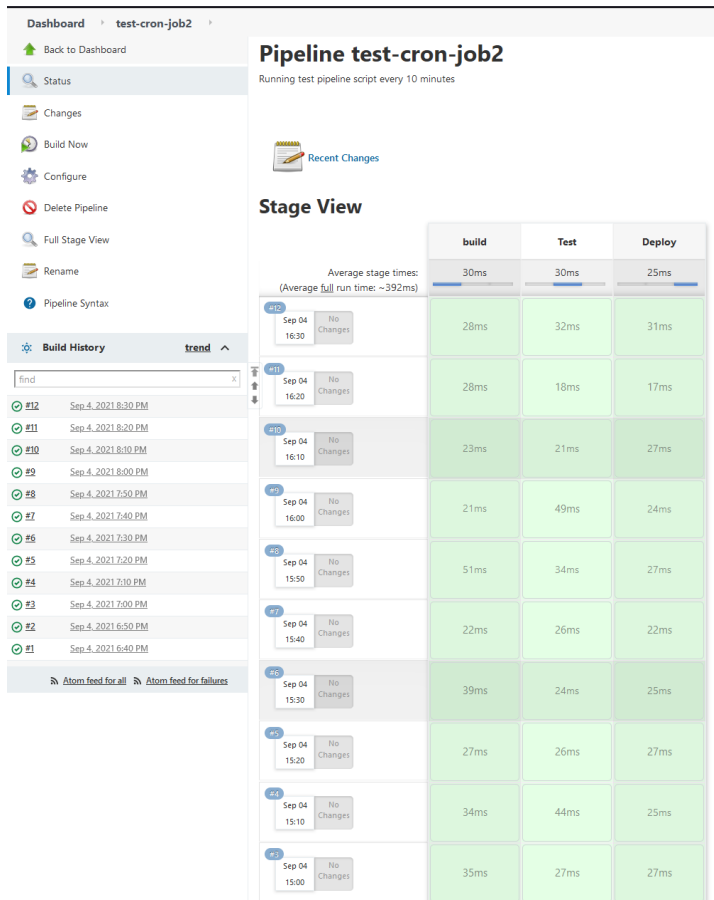
```
*/* 18-20 4 9 *
```

This tells the pipeline to build itself every 10 minutes from 18-20(6pm-8pm) on the 4th of september. For the pipeline script itself, I submitted the following:

```
node{
  stage('build'){
    echo 'this is the build stage'
  }
  stage('Test'){
    echo 'this is the test stage'
  }
  stage('Deploy'){
    echo 'this is the deploy stage'
  }
}
```

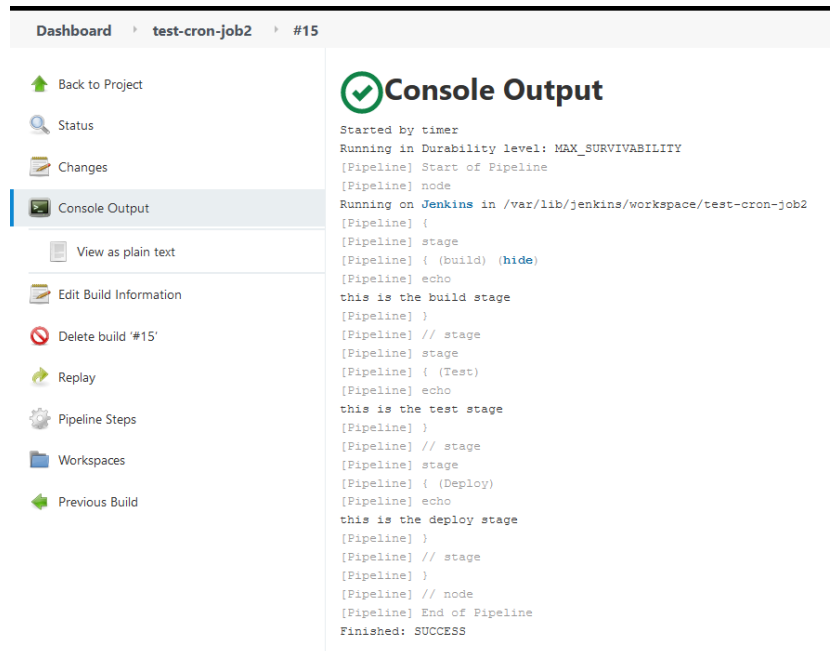
Each echo was the indicator that each stage was executed without issue.

The result of the job was successful and it performed as expected. Below are screenshots of the stages and an example console output from one of the builds. They all had the same outputs



The screenshot shows the Jenkins interface for the pipeline 'test-cron-job2'. The left sidebar contains navigation links: Dashboard, Back to Dashboard, Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Rename, and Pipeline Syntax. The main area displays the 'Stage View' for the pipeline, which is running every 10 minutes. A table shows the execution times for the 'build', 'Test', and 'Deploy' stages across multiple builds. The table includes columns for the build number, the stage name, and the execution time in milliseconds. The average stage times are 30ms for build, 30ms for Test, and 25ms for Deploy. The average full run time is approximately 392ms.

Build	build	Test	Deploy
#12	28ms	32ms	31ms
#11	28ms	18ms	17ms
#10	23ms	21ms	27ms
#9	21ms	49ms	24ms
#8	51ms	34ms	27ms
#7	22ms	26ms	22ms
#6	39ms	24ms	25ms
#5	27ms	26ms	27ms
#4	34ms	44ms	25ms
#3	35ms	27ms	27ms



The screenshot shows the Jenkins interface for the pipeline 'test-cron-job2' at build #15. The left sidebar contains navigation links: Dashboard, Back to Project, Status, Changes, Console Output, View as plain text, Edit Build Information, Delete build '#15', Replay, Pipeline Steps, Workspaces, and Previous Build. The main area displays the 'Console Output' for the build, which shows the execution of the pipeline script. The output includes the following text:

```
Started by timer
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/test-cron-job2
[Pipeline] {
[Pipeline] stage
[Pipeline] { (build)
[Pipeline] echo
this is the build stage
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] echo
this is the test stage
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] echo
this is the deploy stage
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] End of Pipeline
Finished: SUCCESS
```

## Shutting down EC2 Instance

Prior to proceeding, please check the following:

- Aws configure
- Timezone

For AWS configurations, you can use

```
aws configure
```

Check to see if your keys and region are correct.

For timezone, you can use

```
timedatectl list-timezones
```

for the list of timezones that aws has available. And see which one is most appropriate for you. Then you can update your timezone using

```
sudo timedatectl set-timezone <your timezone here>
```

These were the solutions for issues I had faced.

To shutdown the EC2 Instance, I opened my EC2 on my terminal. The command

```
Sudo shutdown now -h
```

will stop the current instance. I then created a bash script I was going to use to shut down the EC2 Instance inputting the above command. To do this I used

```
Nano ec2_shutdown.sh
```

This will create the bash file ec2\_shutdown.sh in the current directory. Then typed in

```
#!/usr/bin/bash
```

```
Sudo shutdown now -h
```

With this, the script to shutdown the EC2 was made. I wanted to shut down the EC2 instance by triggering the ec2\_shutdown.sh script I just made by the end of a weekday class (around 9pm or 21:00).

I checked my current directory with the bash file in it by using pwd. In my case it was /home/ec2-user/.

Now I am able to use

```
Crontab -e
```

To edit the details of a scheduled job. In this case i wanted to schedule a trigger of ec2\_shutdown.sh by 9pm so the syntax to achieve this was

```
0 19 * * * sh /home/ec2-user/ec2_shutdown.sh
```

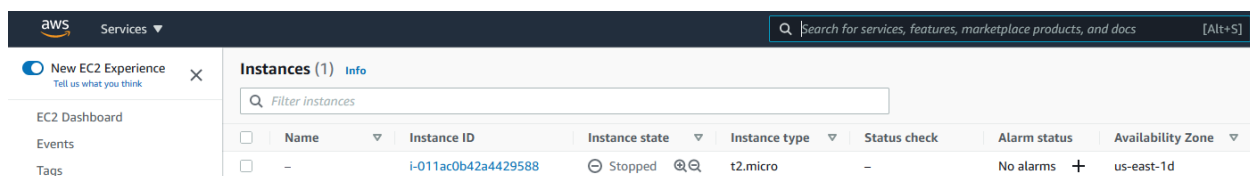
Hitting the esc key and typing

```
:wq
```

I have saved this cronjob.

To check if this job ran at the scheduled time, the instance should return the below message and the instance status should show stopped on the AWS EC2 web interface.

```
[ec2-user@ip-172-31-89-52 ~]$ Connection to ec2-52-91-176-30.compute-1.amazonaws.com closed by remote host.
Connection to ec2-52-91-176-30.compute-1.amazonaws.com closed.
```



The screenshot shows the AWS Management Console interface. At the top, there's a search bar and a navigation menu. The main content area displays the 'Instances (1)' page. A table lists the instance details:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
-	i-011ac0b42a4429588	Stopped	t2.micro	-	No alarms	us-east-1d