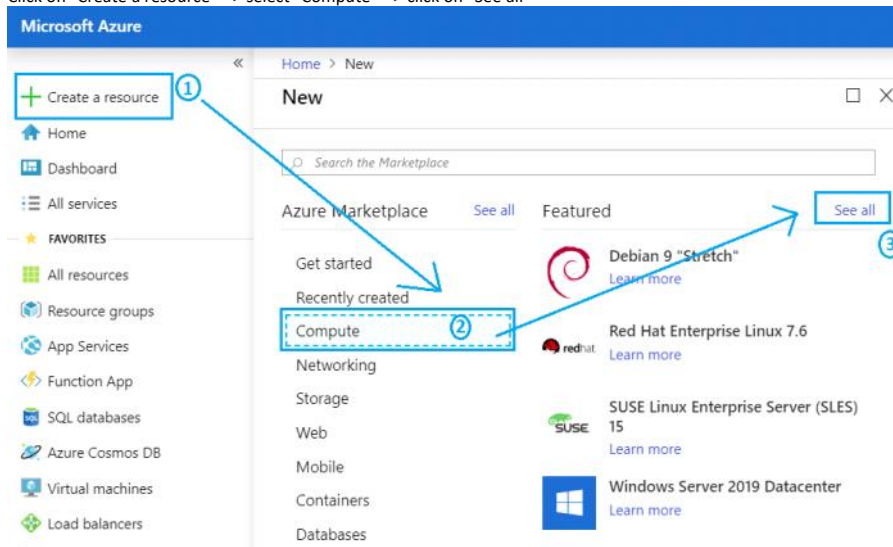


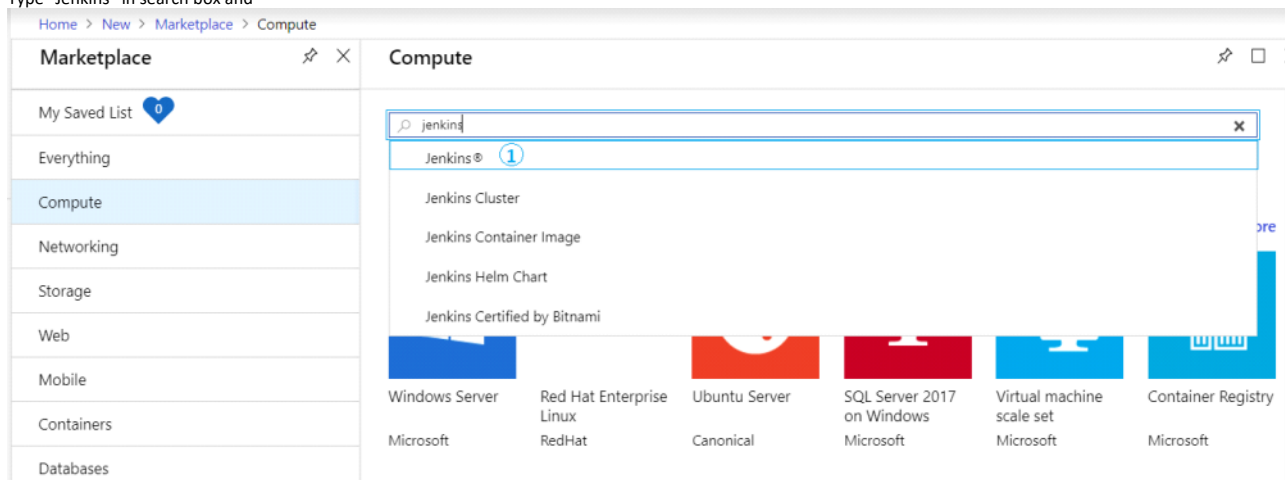
Provisioning an Azure Jenkins VM

Tuesday, April 16, 2019 10:58 AM

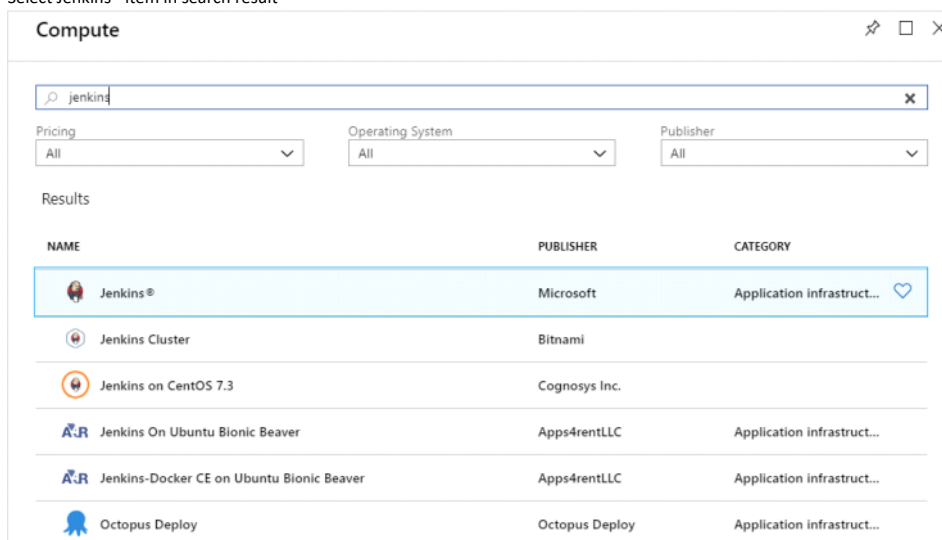
1. Using your account to access to Azure portal <https://portal.azure.com/>. If you do not have one, you can sign up for a [free trial](#)
2. Click on "Create a resource" --> select "Compute" --> click on "See all"



3. Type "Jenkins" in search box and



Select Jenkins® item in search result




5. An Azure [Marketplace](#) > [Compute](#) > [Jenkins®](#) will be displayed, click on "Create" button to create

Home > Compute > Jenkins®

Jenkins®

Microsoft

 **Jenkins®**
Microsoft

[Create](#) [Save for later](#)

We are excited to bring the next phase of our support for Jenkins on Microsoft Azure with the launch of a secure, stable and production ready version of Jenkins.

Note: For instructions on connecting to this Jenkins instance once deployed, please browse to the URL or public IP of this instance. The URL is the Domain name label you enter in Settings and the suffix shown below this field.


This solution template will install the latest stable Jenkins version on a Linux (Ubuntu 16.04 LTS) VM along with tools and plugins configured to work with Azure. This includes –

- git for source control
- Azure Credentials plugin for connecting securely
- Azure VM and Container Agents plugin for elastic build, test and continuous integration
- Azure Storage plugin for storing artifacts
- Azure CLI to deploy apps using scripts
- Azure plugins to make it easy for you to deploy to App Service, Function, AKS, Service Fabric and VMSS.

For a detailed walkthrough of the steps this solution automates for you, please visit this [quickstart](#).

Jenkins® is a registered trademark of Software in the Public Interest, Inc.

Useful Links
[Jenkins on Azure Developer Hub for quickstarts and tutorials](#)
[Need to optimize Jenkins? Experiencing a technical issue? Sign up for CloudBees Jenkins Support and get help from the #1 Jenkins sponsor](#)



6. The Configure basic settings screen will be display and input your VM Name, Username/password, create new resource group

Home > Get Started > Jenkins® > Create Jenkins® > Basics

Create Jenkins®

Basics

1 Basics
Configure basic settings

2 Additional Settings

3 Integration Settings

4 Summary
Jenkins®

5 Buy

Basics

* Name
jenkinsjavaCICD ✓

* User name
vm_admin ✓

* Authentication type
Password SSH public key

* Password
..... ✓

* Confirm password
.....

Subscription
Visual Studio Enterprise

* Resource group ⓘ
Select existing...
[Create new](#)

A resource group is a container that holds related resources for an Azure solution.

* Name
.....

OK Cancel

7. Click on button "OK"

Home > Get Started > Jenkins® > Create Jenkins® > Basics

Create Jenkins®

- 1 Basics
Configure basic settings
- 2 Additional Settings
- 3 Integration Settings
- 4 Summary
Jenkins®
- 5 Buy

Basics

* Name
jenkinsJavaCICD ✓

* User name
vm_admin ✓

* Authentication type
Password SSH public key

* Password
..... ✓

* Confirm password
..... ✓

Subscription
Visual Studio Enterprise

* Resource group ⓘ
(New) DevOps
[Create new](#)

* Location
Southeast Asia

OK

8. Additional Settings screen will be displayed
 Select your appropriate Size or click on "Change size" to change into other VM size that you want to
 Type the unique "Domain Name label" that does not exist with other
 For the rest other leave it by default

Home > Get Started > Jenkins® > Create Jenkins® > Additional Settings > Subnets

Create Jenkins®

- 1 Basics
Done ✓
- 2 Additional Settings
- 3 Integration Settings
- 4 Summary
Jenkins®
- 5 Buy

Additional Settings

* Size
1x Standard D52 v2
2 vcpus, 7 GB memory
[Change size](#)

VM disk type ⓘ
SSD HDD

* Virtual network
(new) jenkins-vnet
Subnets
Configure subnets ⓘ

* Public IP address ⓘ
(new) jenkins-pip

* Domain name label
vtsjenkin0001 ✓
southeastasia.cloudapp.azure.com

Jenkins rel
LTS
JDK to be installed to the Jenkins instance. Default is Azul System Zulu Enterprise Builds of OpenJDK.

JDK Type ⓘ
Zulu
Zulu
OpenJDK

Subnets

* Subnet name
jenkins ✓

* Subnet address prefix
10.1.0.0/24 ✓

Click "OK"

9. The Integration Settings screen will be displayed. Leave all items as default and click on button "OK" to continues

1 Basics Done ✓

2 Additional Settings Done ✓

3 Integration Settings >

Service Principal Integration

Auto(MSI)

Add a default cloud template for agents. ACI: Azure Container Instance, VM: virtual machine.

Enable Cloud Agents

No ACI VM

OK

10. The Summary will be displayed to validate all previous step, if everything okay the "OK" button will be enabled and you should click on "OK" to continue
11. The last screen will be display to require you accept the condition and term ...
12. Click on "Create" button to start provisioning Jenkins VM

Create Jenkins®

1 Basics Done ✓

2 Additional Settings Done ✓

3 Integration Settings Done ✓

4 Summary Jenkins® ✓

5 Buy >

Create

Jenkins®

by Microsoft

Terms of use | privacy policy

Deploying this template will result in various actions being performed, which may include the deployment of one or more Azure resources or Marketplace offerings and/or transmission of the information you provided as part of the deployment process to one or more parties, as specified in the template. You are responsible for reviewing the text of the template to determine which actions will be performed and which resources or offerings will be deployed, and for locating and reviewing the pricing and legal terms associated with those resources or offerings.

Current retail prices for Azure resources are set forth [here](#) and may not reflect discounts applicable to your Azure subscription.

Prices for Marketplace offerings are set forth [here](#), and the legal terms associated with any Marketplace offering may be found in the Azure portal; both are subject to change at any time prior to deployment.

Neither subscription credits nor monetary commitment funds may be used to purchase non-Microsoft offerings. These purchases are billed separately. If any Microsoft products are included in a Marketplace offering (e.g., Windows Server or SQL Server), such products are licensed by Microsoft and not by any third party.

Template deployment is intended for advanced users only. If you are uncertain which actions will be performed by this template, which resources or offerings will be deployed, or what prices or legal terms pertain to those resources or offerings, do not deploy this template.

Terms of use

By clicking "Create", I (a) agree to the legal terms and privacy statement(s) provided above as well as the legal terms and privacy statement(s) associated with each Marketplace offering that will be deployed using this template, if any; (b) authorize Microsoft to charge or bill my current payment method for the fees associated with my use of the offering(s), including applicable taxes, with the same billing frequency as my Azure subscription, until I discontinue use of the offering(s); and (c) agree that Microsoft may share my contact information and transaction details with any third-party sellers of the offering(s). Microsoft assumes no responsibility for any actions performed by third-party templates and does not provide rights for third-party products or services. See the [Azure Marketplace Terms](#) for additional terms.

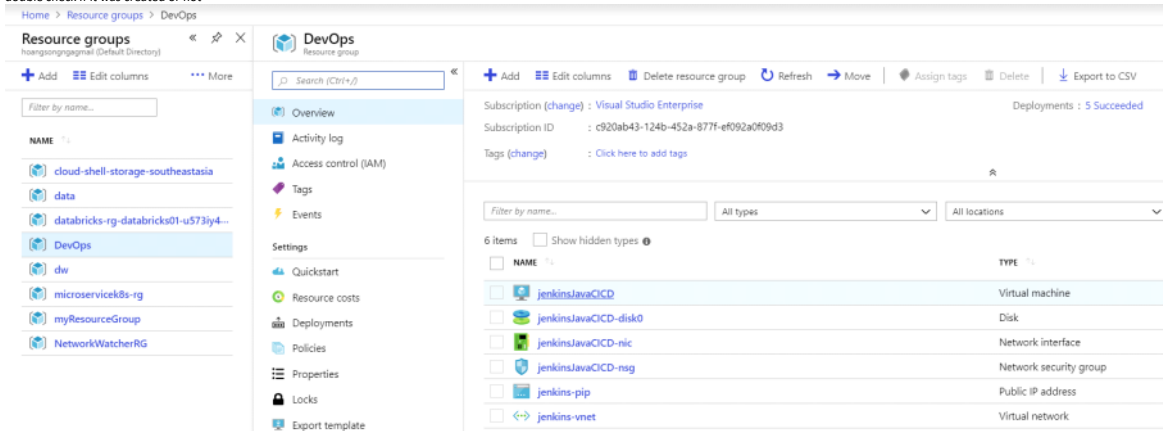
☒ I give Microsoft permission to use and share my contact information so that Microsoft or the Provider can contact me regarding this product and related products.

Create

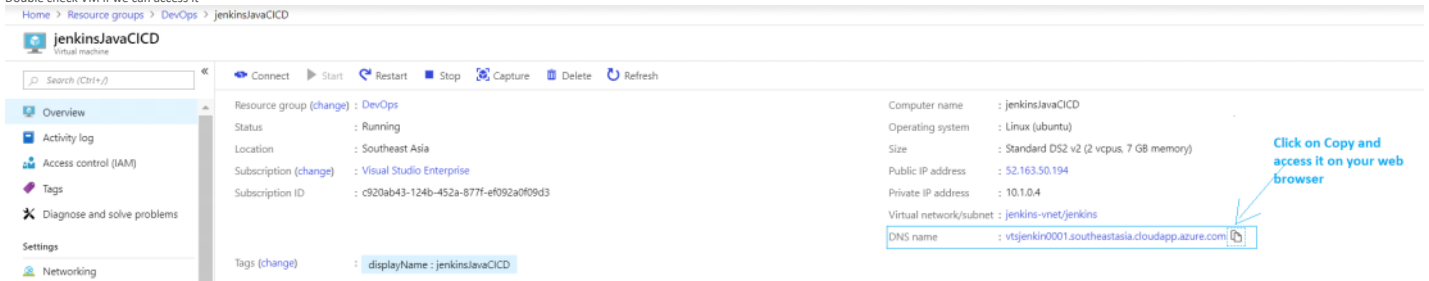
Setting up the Jenkins VM

Tuesday, April 16, 2019 11:47 AM

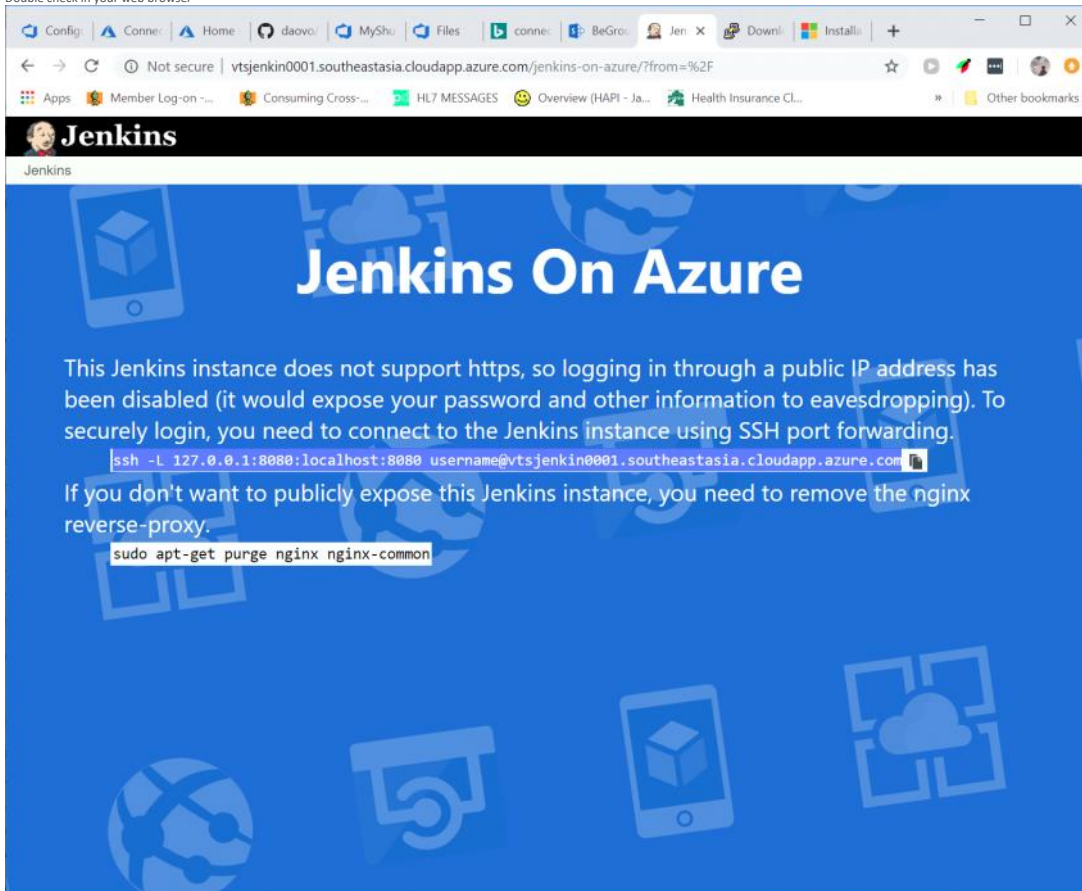
1. Once the Jenkins VM is provisioned, please go to resource group to find out VM name that you named to double check if it was created or not



2. Double check VM if we can access it



4. Double check in your web browser



5. Using SSH to connect to Jenkins VM



jenkinsJavaCICD
Virtual machine

1: Click on Connect

Resource group (change): DevOps

Status: Running

Location: Southeast Asia

Subscription (change): Visual Studio Enterprise

Subscription ID: c920ab43-124b-452a-877f-ef092a0f0d3

Tags (change): displayName: jenkinsJavaCICD

Computer name: jenkinsJavaCICD

Operating system: Linux (ubuntu)

Size: Standard D52 v2 (2 vcpus, 7 GB memory)

Public IP address: 52.163.50.194

Private IP address: 10.1.0.4

Virtual network/subnet: jenkins-vnet/jenkins

DNS name: vtsjenkin0001.southeastasia.cloudapp.azure.com

Show data for last: 1 hour 6 hours 12 hours 1 day 7 days 30 days

CPU (average)

Percentage CPU (Avg) jenkinsJavaCICD: 0.64%

Network (total)

Network In Total (Sum) jenkinsJavaCICD: --

Network Out Total (Sum) jenkinsJavaCICD: --

Disk bytes (total)

Disk Read Bytes (Sum) jenkinsJavaCICD: 348.08 KB

Disk Write Bytes (Sum) jenkinsJavaCICD: 53.06 KB

Connect to virtual machine

To improve security, enable just-in-time access on this VM.

RDP SSH

To connect to your virtual machine via SSH, select an IP address, optionally change the port number, and use one of the following commands:

* IP address: DNS name (vtsjenkin0001.southeastasia.cloudapp.azure.com)

* Port number: 22

Login using VM local account: 2: Copy

Uh vm_admin@vtsjenkin0001.southeastasia.cloudapp.azure.com

Inbound traffic to the Public IP address may be blocked. You can update inbound port rules in the **VM Networking** page.

You can troubleshoot VM connection issues by opening the **Diagnose and solve problems** page.

6. To initiate an SSH tunnel, the following command needs to be run from a Command Prompt. An SSH tunnel creates a secure connection between your host and remote computer through which services can be relayed. If this command is successful, you should be able to access the remote Jenkins on port 8080 on your local machine.

```
putty.exe -ssh -L 8080:localhost:8080 <username>@<ip address>
```

Open your CMD from Run on window dialog to access to the VM by using your username/password from previous steps

```
vm_admin@jenkinsJavaCICD: ~
C:\Users\trvo>ssh -L 127.0.0.1:8080:localhost:8080 vm_admin@vtsjenkin0001.southeastasia.cloudapp.azure.com
vm_admin@vtsjenkin0001.southeastasia.cloudapp.azure.com's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-1041-azure x86_64)
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-1041-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

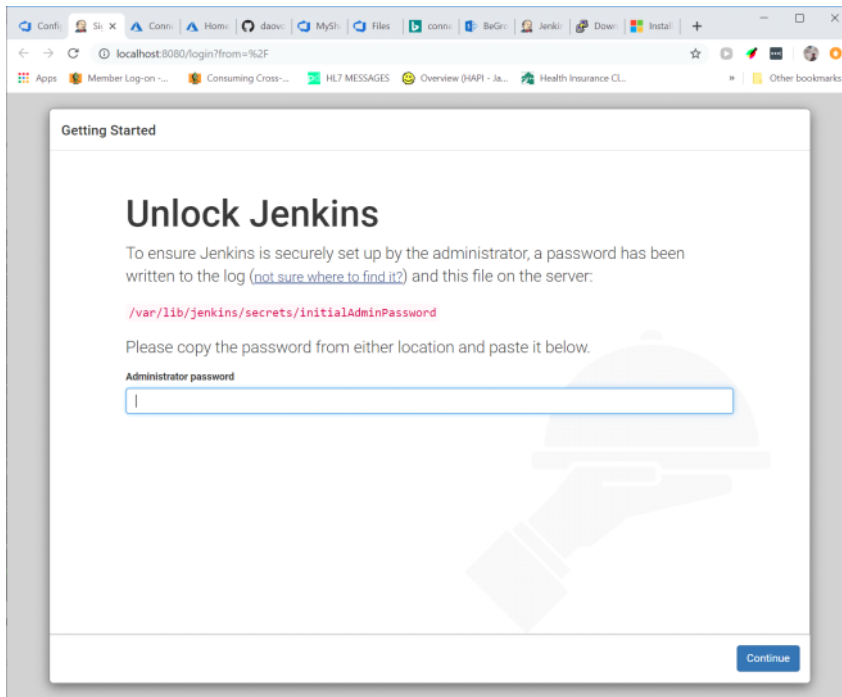
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

vm_admin@jenkinsJavaCICD: ~$
```

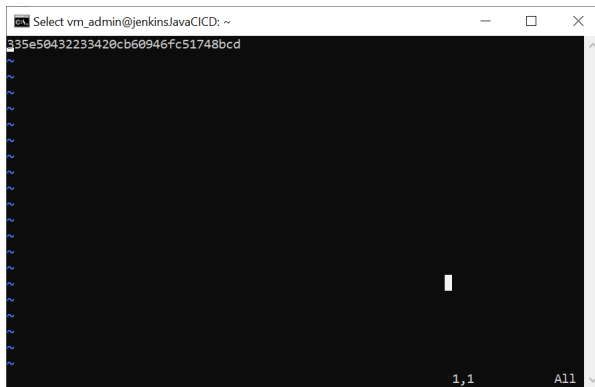
1. Once the connection is successful, open a browser on the host machine and navigate to the URL <http://localhost:8080>. The Getting Started page for Jenkins will be displayed.

2.



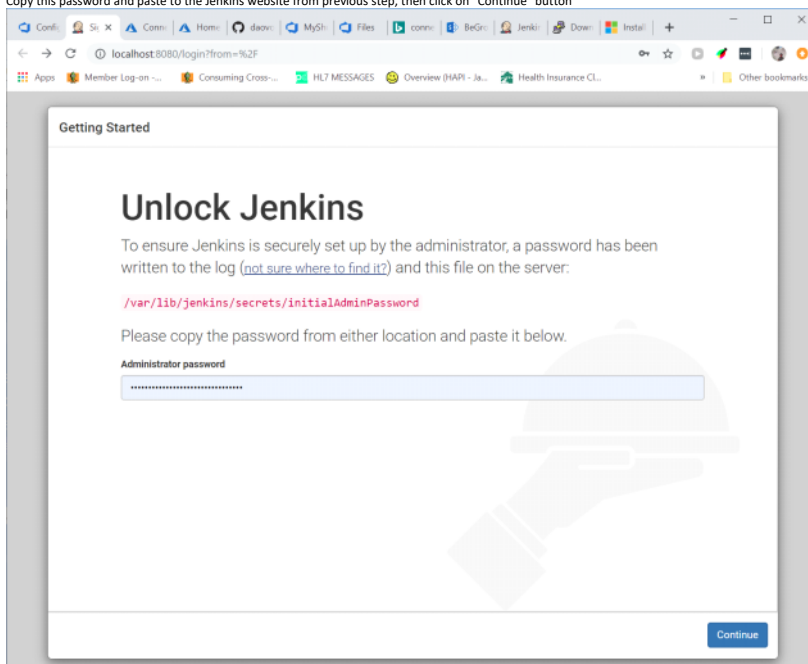
3. For security reasons, Jenkins generates an initial password and save it in a file on the server. This password will need to be retrieved and provided to unlock Jenkins. Return to the **Putty** terminal and type the following command to open the password file and copy the password. You can double click on the password text and use **CTRL+C** to copy the text and place it in the clipboard. Press the **Esc** button and then type `:q!` at the prompt to exit the vi editor without saving the file.

`sudo vi /var/lib/jenkins/secrets/initialAdminPassword`

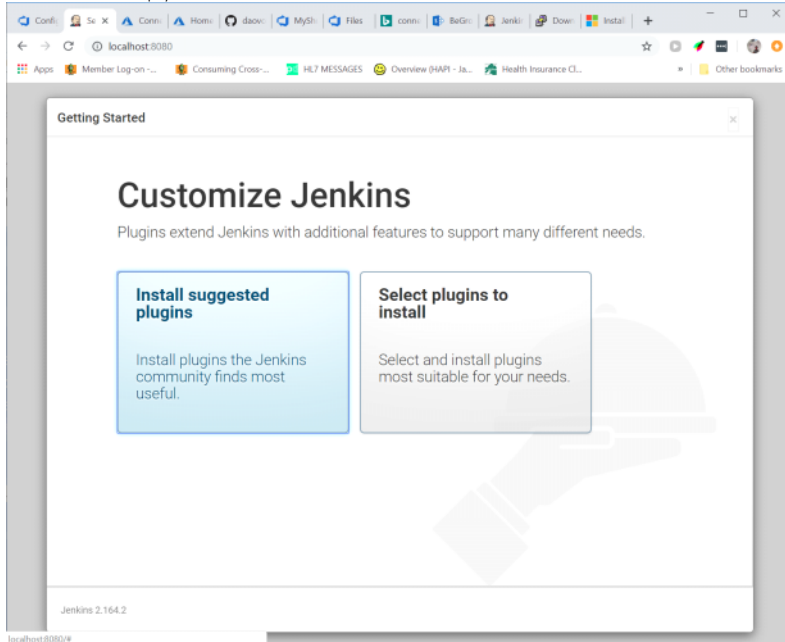


4. Copy this password and paste to the Jenkins website from previous step, then click on "Continue" button

5.

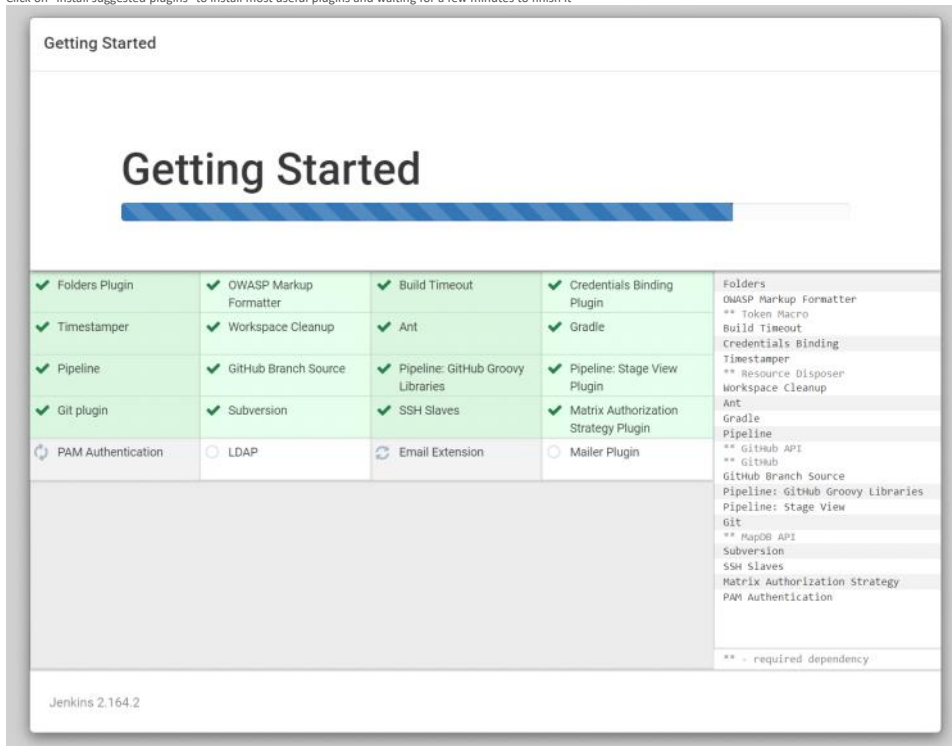


6. The browser should be displayed



7.

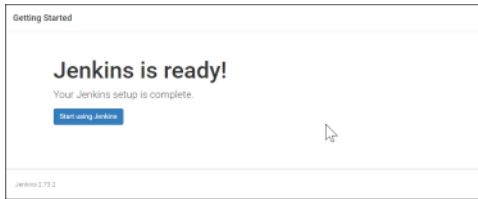
8. Click on "Install suggested plugins" to install most useful plugins and waiting for a few minutes to finish it



9.

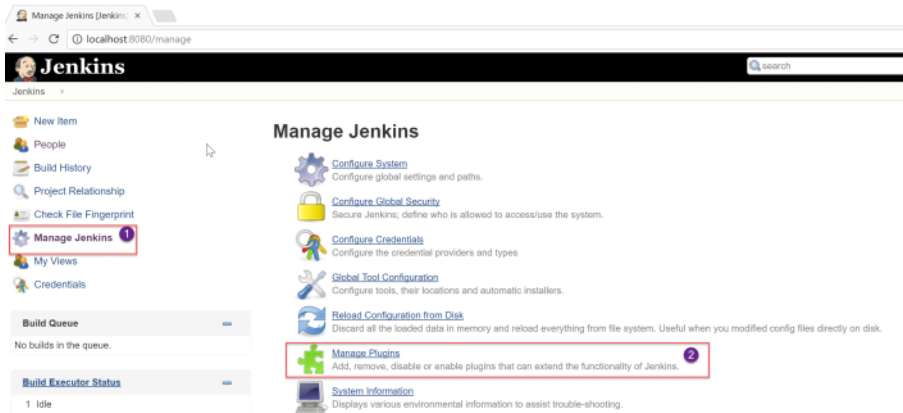
10. The final step is to create a new Admin user. Provide User name, Password, Full name and Email address for the user. Select **Save and Finish** when you are done.

11. Jenkins is now ready for use. Select **Start using Jenkins** to start using it.

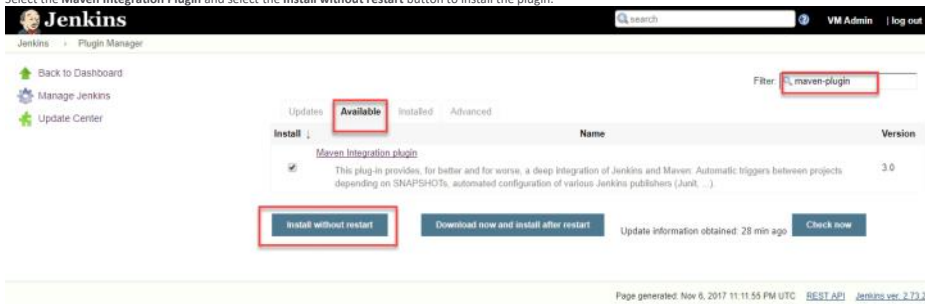


Installing and Configuring Plugins

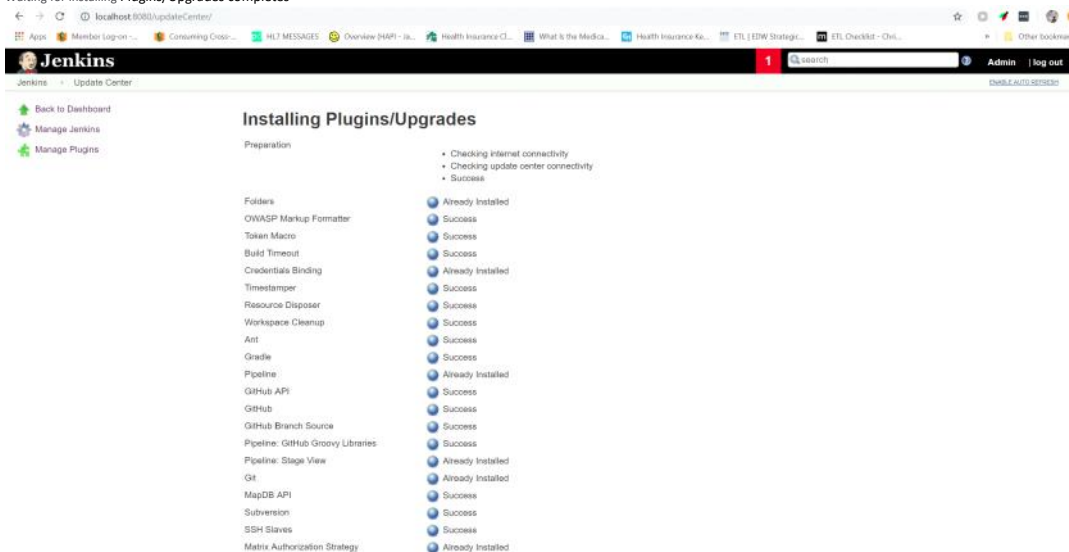
1. We will now install the Maven and VSTS (yet to be renamed Azure DevOps!) plugins that we require for this lab. Click **Manage Jenkins** on the Jenkins home page and select **Manage Plugins**. Select the **Available** tab and search for **team services**



2. Select VS Team Services Continuous Deployment/ GitHub Authentication/ GitLab Authentication plugin and select **Install without restart**
3. Select **Manage Plugins**, select the **Available** tab and search for **maven-plugin**
4. Select the **Maven Integration Plugin** and select the **Install without restart** button to install the plugin.

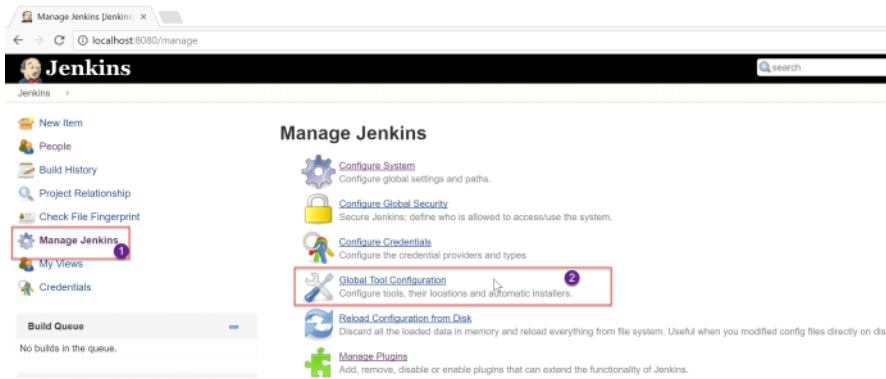


5. Waiting for installing **Plugins/Upgrades** completes



6.

7. Once the plugin is installed, go back to **Manage Jenkins** and select the **Global Tool Configuration** option.



Note: Jenkins provides great out-of-the-box support for Maven. Since Maven is not yet installed, it can be manually installed by extracting the tar file located in a shared folder. Alternatively, when the **Install automatically** option is selected in the **Global Tool Configuration** screen, Jenkins will download and install Maven from the Apache website when a build job requires it.

- To install Maven, select the **Install automatically** option and select the **Apply** button. The latest version of Maven at the time of writing this lab is 3.5.4 (latest version is 3.6.1)

Maven

Maven Installations

Maven Name

☒ Install automatically

☐ Install from Apache

Version

- Select the **Back to Dashboard** button to return to the home page. We are done with the setup. Let's go and create a new CI job.

Creating a new build job in Jenkins

- From the Jenkins home page, click the **New Item** link. Provide a name for the build definition, select **Maven** project and click **OK**.

Jenkins

Enter an item name

* Required field

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

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as 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
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 folders.

☒ OK

- Now scroll down to the **Source code Management** section. Select **Git** and provide the clone URL of the Azure DevOps Git repo in the format [https://dev.azure.com/your org name/\(team project name\)/_git/MyShuttle](https://dev.azure.com/your org name/(team project name)/_git/MyShuttle) (or github or gitlab url but my suggestion we should use dev.azure.com repos). If you do not see **Git** under Source code management (not a usual thing), you will need to install/enable the Git plugin.

Jenkins > MyShuttle

General **Source Code Management** Build Triggers Build Environment Pre Steps Build Post Steps

Build Settings Post-build Actions

Source Code Management

☐ None ☒ Git

Repositories

Repository URL

Failed to connect to repository: Command "git ls-remote -h https://dev.azure.com/myshuttle/_git/MyShuttle HEAD" returned status code 128: stdout: stderr: fatal: Authentication failed for 'https://dev.azure.com/myshuttle/_git/MyShuttle'

Credentials

Advanced... Add Repository

Branches to build

Branch Specifier (blank for 'any')

- Your Azure repo is very likely to be private. Unless you have a public repo, you should provide the credentials to access the repository. If you do not have one or don't remember the credentials, go to your Azure Repos and select the **Clone** option. Select **Generate Credentials** and enter a **User name** and **Password**. Click **Save Git Credentials** to save.

Clone repository

Clone Git repository using command line or IDE

Command line

HTTPS SSH

https://[redacted]_git/MyShuttle/...

User name (primary)
sachin

Alias (optional)
sachin77

Password *

Confirm Password *

Save Git Credentials

[Create a Personal access token](#)
[Learn more about authentication options](#)

IDE

Clone in VS Code

Having problems authenticating in Git? Be sure to get the latest version of [Git for Windows](#) or our plugins for [IntelliJ](#), [Eclipse](#), [Android Studio](#) or [Windows command line](#).

- Select the **Add | Jenkins** option to add a new credential. Provide the **User name** and **Password** created earlier and click the **Add** button to close the wizard

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain: Global credentials (unrestricted)

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: vmadmin

Password: [redacted]

ID:

Description:

Add Cancel

- Select the credential created in the previous step from the drop-down. The error message should disappear.

General **Source Code Management** Build Triggers Build Environment Pre Steps Build Post Steps Build Settings

Post-build Actions

Source Code Management

None Git

Repositories

Repository URL: [redacted]_git/MyShuttle

Failed to connect to repository : Command "git ls-remote -h [redacted]_git/MyShuttle HEAD" returned status code 128:
stdout:
stderr: fatal: Authentication failed for [redacted]_git/MyShuttle"

Credentials: [redacted] Add

Advanced...

Add Repository

Save Apply

Branch Specifier (blank for 'any') */master

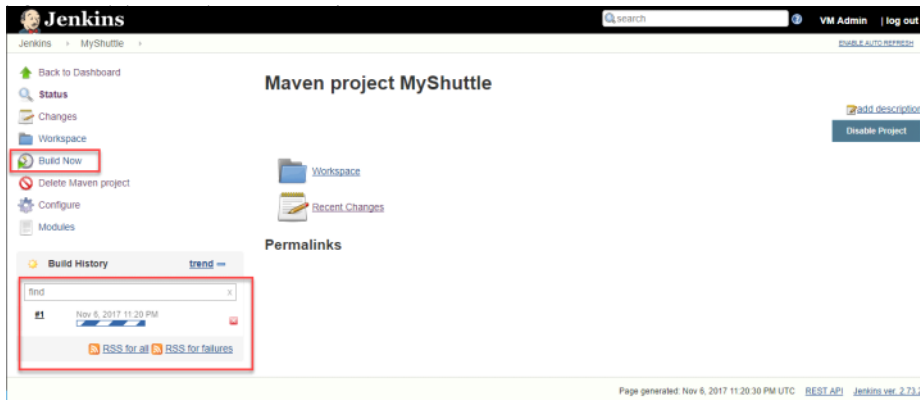
- The source code for this application has both unit tests and UI tests. We are not ready to run the UI test at this point. So, we will specify to run only the unit tests. Scroll down to the **Build** section and provide the text `package -Dtest=FaresTest,SimpleTest` in the **Goals and options** field.

7. Once the build is complete, you can specify what action you want to take. For instance, you can archive the build artifacts, trigger an Azure CD pipeline, deploy directly to Azure App Service, etc., We will choose the **Archive the artifacts** option in the **Post-build Actions**.

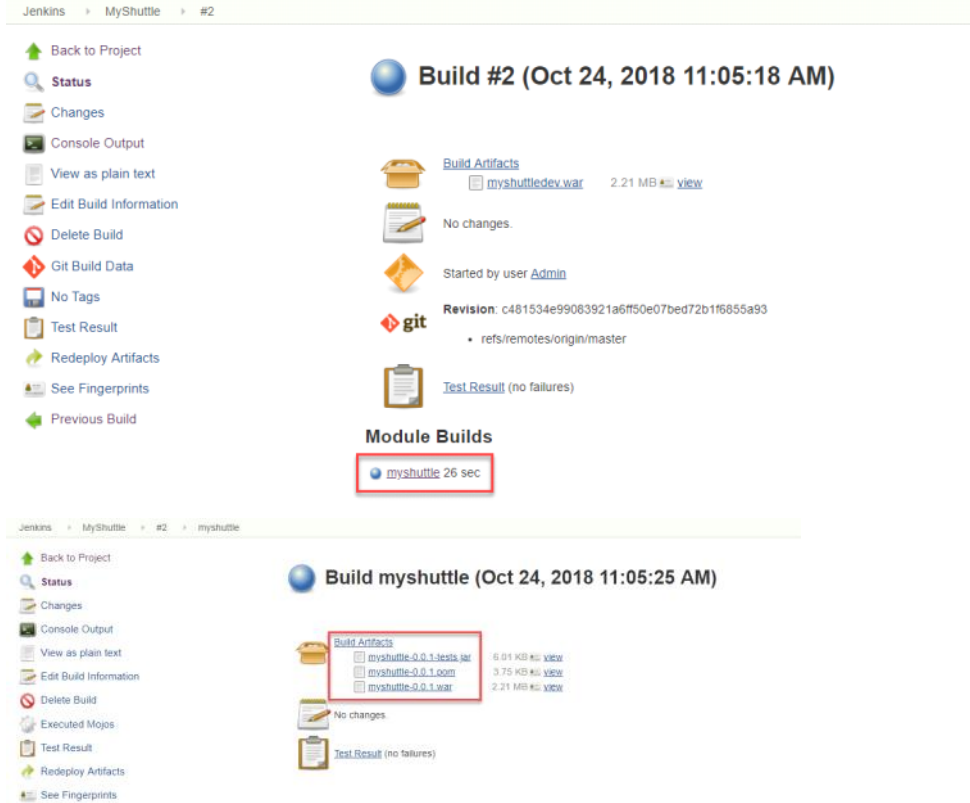
Note: Note there is also **Post-build steps** section which is very similar to the actions section. The tasks configured in the post-build steps/actions are executed after all the build steps have been executed.

8. Enter `target/*.war` in the **Files to archive** text box. Select the **Save** button to save the settings and return to the project page.

9. The configuration is now completed, Select the **Build Now** option to initiate an Ad-hoc build. The build progress will be displayed on the left pane in the **Build History** section



10. To view the build details and the list of build artifacts, select the build number displayed in the **Module Builds** section.



11. Select the **Test Result** option to view the results of the unit tests that were included in the build definition. Next, we will explore the two different options available to trigger the Jenkins CI job when a code is pushed to Azure Repos.

Approach 1: Triggering the CI via a service hook in Azure DevOps

In this approach, a service hook will be configured in Azure DevOps to trigger a Jenkins build upon a code commit. Service hooks enable you to perform tasks on other services when events happen in your Azure DevOps Services projects.

1. To configure the service hook, navigate to the Azure DevOps project settings page and select **Service hooks** under **General**. Select **+ Create subscription**.
2. In the **New Service Hook Subscriptions** screen, select the **Jenkins** option and then click the **Next** button. Jenkins service supports three events - **Build completed**, **Code Pushed** and **Pull request merged**. We are only interested in the code push event - so, select **Code pushed** for the **Trigger on this type of event** field. Select the **MyShuttle** repository and then click **Next**.

X

Select and configure the action to perform.

Perform this action


Trigger Git build

Triggers a build configured to use a Git repository using the [Jenkins Git Plugin](#). Secure, HTTPS endpoints are recommended due to the potential for private data in the event payload.

Jenkins base URL ⓘ

Jenkins base URL required

User name  required

vmadmin 

User API token (or password) required

Integration level optional

DevOps plugin for Jenkins ✓

[Previous](#)

Next

Test

Finish

Cancel

3. Provide the following details in the **Select and configure the action to perform** screen
 1. Select the **Trigger Git build option**
 2. Provide the **Jenkins base URL** in [http://\[ip address or the host name\]](#) format
 3. Provide the **User name** and **Password** to trigger the build. Note the username and password is the credentials of the Jenkins administrator user that you configured earlier
4. Click the **Test** button to validate the configuration and then click **Finish**. This will set the trigger to initiate the Jenkins CI build whenever a source code change is committed on the repository.
5. Try making a commit to the code. [src/main/webapp/index.jsp](#) would be a good candidate. This should trigger the MyShuttle build on Jenkins. You can confirm it by checking the history tab of the Jenkins services hngk

Project Settings > Service hooks

General

Overview

Services

Teams

Security

Notifications

Service hooks

Dashboards

Service Hooks

Integrate with your favorite services by notifying them when events happen in your project.

+

✎

🔄

✖

History

Consumer ↑

Event ↑

Action

Jenkins

*** Code pushed

Repository MyShuttle

Trigger Git build

Server: 52.183.0.156

HISTORY

Code pushed > Jenkins (Trigger Git build)

Created by: Sachin Raj (10 minutes ago)

Modified by: Sachin Raj (10 minutes ago)

Detailed

Logging: On

🔄

🔍

⌵

Summary

Request

Response

Event

Status	When ↑	
✔ Succeeded	9 minutes ago	Sent at: Friday, September 28, 2018 8:35:09 PM
✔ Succeeded	2 minutes ago	<div>Message</div> <div>Sachin Raj pushed updates to MyShuttle/master</div>

⏮

⏭

Last 7 days

2 succeeded

Close

In this approach, Jenkins CI job will be nested within an Azure CI pipeline. The key benefit of this approach is you can have end-to-end traceability from work items to source code to build and release pipelines. To begin, an endpoint to the Jenkins Server for communication with Azure DevOps will be configured.

1. Go to your project settings. Select **Pipelines** and **Service connections**, click **New service connection** and choose **Jenkins** from the dropdown.
2. Provide a connection name, Jenkins server URL in the format [http://\[server IP address or DNS name\]](http://[server IP address or DNS name]) and Jenkins user name with password. Select **Verify Connection** and validate the configuration. If it successful, then select **Ok**.

Add Jenkins service connection

Connection name:

Server URL:

Accept untrusted SSL certificates: ☐

Username:

Password:

Connection: ✔ Verified [Verify connection](#)

[OK](#) [Close](#)

The next step would be to configure the build pipeline.

- Go to **Azure Pipelines and Builds**, Click **+New** and select **New build pipeline** to create a new build definition.
- At the time of writing this lab, Azure Pipelines did not support Jenkins in YAML. Select **Use the Visual Designer** to create a pipeline without a YAML.
- Select **MyShuttle** project, repository and click **Continue**.
- Scroll down and select the standard **Jenkins** template Click **Apply**.

Select a template

Or start with an [Empty job](#)

Docker images to be pushed to a container registry.



Azure Web App for Java

Build a Java WAR file and deploy it to an Azure Web App.



C# Function

Build and test a C# (.NET class library) based Azure Function.



Go (preview)

Build a Go application.



Gradle

Build and test a Java project with Gradle.



Jenkins

Queue a Jenkins job and download its artifacts.

[Apply](#)


Load test using Azure IaaS virtual machines

Create a rig on Azure IaaS virtual machines to run load tests using VSTS cloud-based load testing service.

- Select **Hosted VS2017** for the Agent Queue, provide **MyShuttle** as the Job name (name of the build definition that was created in Jenkins) and then select the Jenkins service endpoint created earlier.

MyShuttle-Jenkins-CI

Tasks Variables Triggers Options Retention History [Save & queue](#) [Discard](#) [Summary](#) [Queue](#) [...](#)

Pipeline Build pipeline [View YAML](#)

Get sources [MyShuttle](#) [master](#)

Agent job 1 [Run on agent](#)

Queue Jenkins Job: MyShuttle [Jenkins Queue Job](#)

Download artifacts produced by MyShuttle [Jenkins Download Artifacts](#)

Publish Artifact: drop [Publish Build Artifacts](#)

Name *

Agent pool * [Pool information](#) [Manage](#)

Parameters [Unlink all](#)

Job name *

Jenkins service connection * [Manage](#) [New](#)

- Next, select the **Get Sources** step. Since Jenkins is being used for the build, there is no need to download the source code to the build agent. To skip syncing with the agent, select **Don't sync sources** option.

Pipeline Build pipeline [View YAML](#)

Get sources [MyShuttle](#) [master](#)

Agent job 1 [Run on agent](#)

Queue Jenkins Job: MyShuttle [Jenkins Queue Job](#)

Download artifacts produced by MyShuttle [Jenkins Download Artifacts](#)

Publish Artifact: drop [Publish Build Artifacts](#)

Repository

Default branch for manual and scheduled builds

Clean [false](#)

☒ Report build status [true](#)

☐ Checkout submodules [false](#)

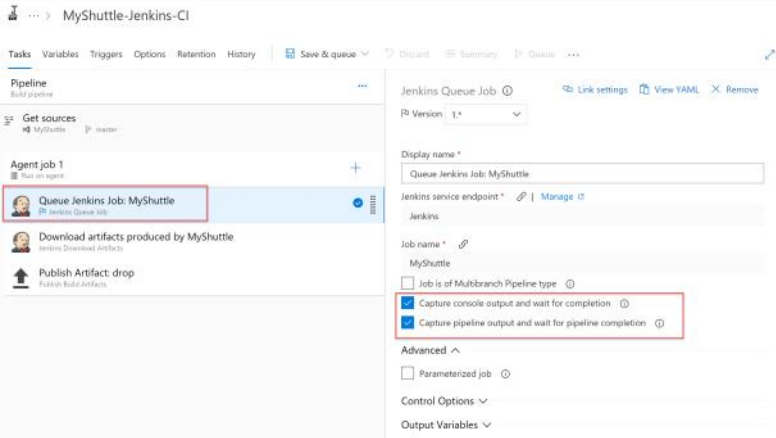
☐ Checkout files from LFS [false](#)

☒ Don't sync sources [true](#)

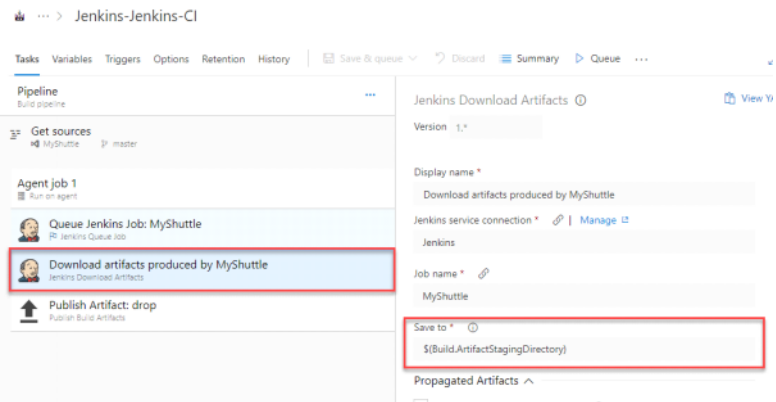
☐ Shallow fetch [false](#)

- Next, select the **Queue Jenkins Job** step. This task queues the job on the Jenkins server. Make sure that the

services endpoint and the job name are correct. The **Capture console output** and the **Capture pipeline output** options available at this step will be selected. The **Capture console output and wait for completion** option, when selected, will capture the output of the Jenkins build console when the Azure build pipeline runs. The build will wait until the Jenkins Job is completed. The **Capture pipeline output and wait for pipeline completion** option is very similar but applies to Jenkins pipelines (a build that has more than one job nested together).



10. The **Jenkins Download Artifacts** task will download the build artifacts from the Jenkins job to the staging directory.

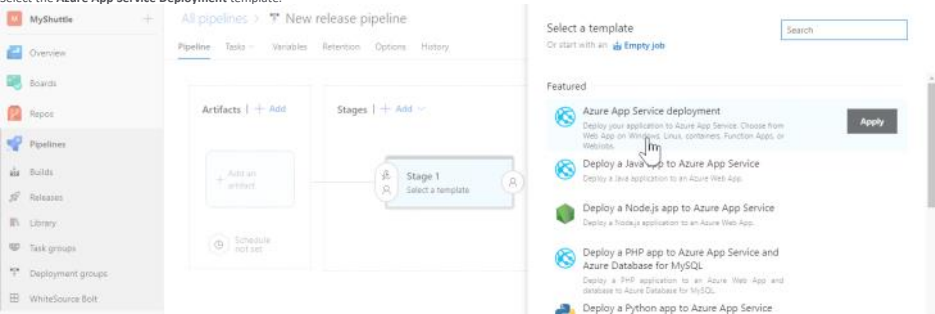


11. The **Publish Artifact drop** will publish to Azure Pipelines.
12. Click **Save & queue** button to save and initiate a new build.

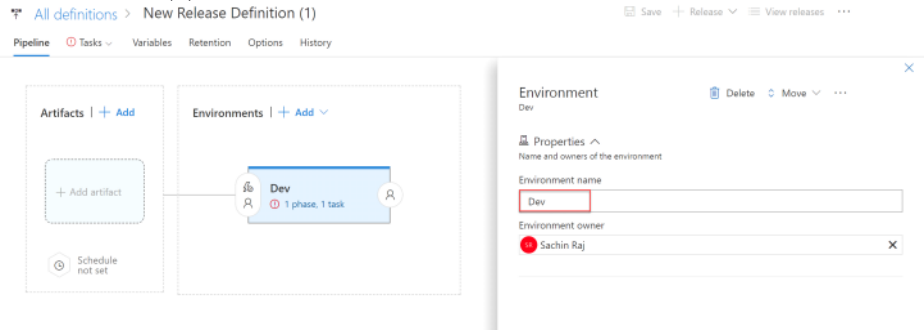
Linking the build artifact for deployment in a CD pipeline

Next, you will configure an Azure CD pipeline to fetch and deploy the artifacts produced by the build. Since the deployment is being done to Azure, an endpoint to Azure will be configured. An endpoint to Jenkins server will also be configured, if not configured earlier.

1. After the endpoint creation, go to the **Releases** tab in **Azure Pipelines**. Open the + drop-down in the list of release pipelines, and choose **Create release pipeline**.
2. Select the **Azure App Service Deployment** template.



3. The default environment for deployment will be named as **Dev**



4. In the **Artifacts** section in the **Pipeline** tab, choose the + **Add** link to select your build artifact.
 1. If you have used the first approach, select **Jenkins** as the **Source type**, select the Jenkins endpoint configured earlier and provide **MyShuttle** for the **Source(job)**, choose the **Default version** as **Latest**. The **Source(job)** should map to the project name configured in Jenkins. If the Jenkins server and the source location is configured correctly, once the publishing of the artifacts is completed, a message with the output file name **myshuttledev.war** will be displayed.

verifylabs / Jenkins / Pipelines

All pipelines > New release pipeline

Pipeline Tasks Variables Retention Options History

Artifacts | + Add 1

Stages | + Add

Dev 1 job, 1 t

Schedule not set

Azure Artifacts Azure Contai... Docker Hub Jenkins 2

Show less ^

Service connection * | Manage 3

Jenkins 3

Jenkins Job * 4

MyShuttle 4

☐ Download artifacts from Azure storage 4

Default version * 5

latest 5

Source alias * 1

_MyShuttle

The artifacts published by each version will be available for deployment in release pipelines. The latest successful build of **MyShuttle** published the following artifacts: **mysuttledev.war**.

Add

2. Otherwise, point this to the Azure CI build pipeline from which the Jenkins CI is executed.
5. Now, the artifact is linked for deployment. Please refer the [Deploying a MySQL Backed Tomcat app on Azure Web App](#) for deploying the WAR file to Azure App Service.

Creating Azure Web App and MySQL database

Tuesday, April 16, 2019 3:23 PM

1. Click on create new the below button to provision a Website app + MySQL Database together to "Create a resource" the same as step "provisioning an Azure Jenkins VM", search Azure Database for MySQL.

Microsoft Azure New > Web App + MySQL > Web App + MySQL > Database Server

Search resources, services and docs

sachin77@outlook.com MICROSOFT (SACHIN77@OUTLO...

Web App + MySQL Create

App name myshuttle-1 ✓

Subscription azurewebsites.net

Windows Azure MSDN - Visual Studio Ultin

Resource Group myshuttle-1 ✓

Create new Use existing

Database Provider Azure Database for MySQL

App Service plan/Location ServicePlanac73d76-8bfe(Centr... >

Database Database Settings Required >

Application Insights On Off

Pin to dashboard

Create Automation options

Database Server

Server name myshuttle-1-mysqldbserver

Server admin login name mysqldbuser

Password ***** ✓

Confirm password ***** ✓

Version 5.7

Pricing tier Standard, 100 Compute Units, 12... >

Database name mysqldbserver55543

OK

2. Wait for the Web App and the database to be provisioned. It roughly takes 3-5 minutes.
3. Navigate to the resource group that you have created. You should see a **Azure Database for MySQL** server provisioned. Select the database server.

Microsoft Azure Resource groups > myshuttle-1

myshuttle-1 Resource group

Search (Ctrl+/)

Overview

Activity log

Access control (IAM)

Tags

SETTINGS

Quickstart

Resource costs

Deployments

Policies

Properties

Locks

Automation script

MONITORING

Metrics

More services >

Subscription (change) Windows Azure MSDN - Visual Studio Ultimate

Deployments 1 Succeeded

Subscription ID 3b-7d76-8bfe-8bfe-101111111111

Filter by name... All types All locations No

3 items

NAME	TYPE	LOCATION
myshuttle-1	App Service	Central US
myshuttle-1-mysqldbserver	Azure Database for MySQL serv...	West US
ServicePlanac73d76-8bfe	App Service plan	Central US

Properties

Subscription (change) Windows Azure MSDN - Visual Studio Ultimate

Deployments 1 Succeeded

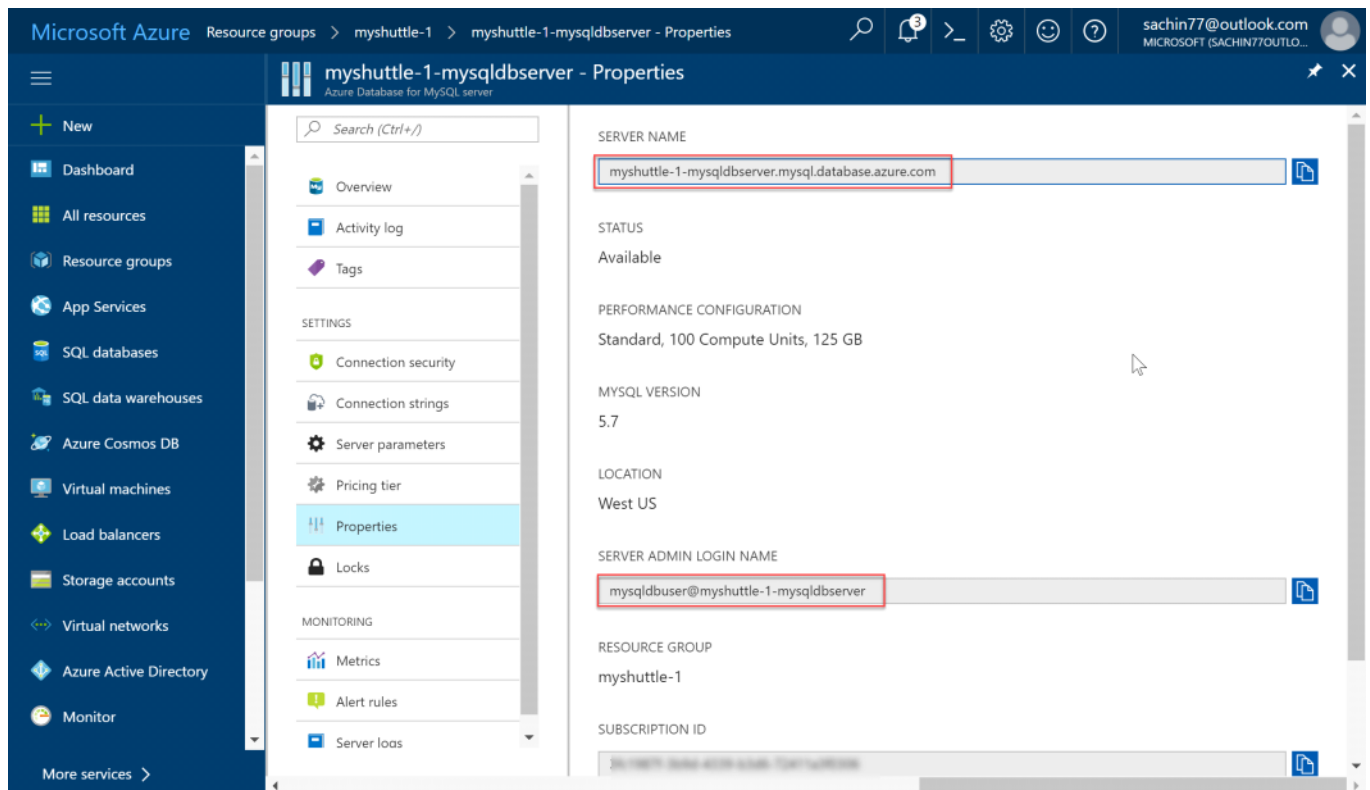
Subscription ID 3b-7d76-8bfe-8bfe-101111111111

Filter by name... All types All locations No

3 items

NAME	TYPE	LOCATION
myshuttle-1	App Service	Central US
myshuttle-1-mysqldbserver	Azure Database for MySQL serv...	West US
ServicePlanac73d76-8bfe	App Service plan	Central US

4. Select **Properties**. Save the **SERVER NAME** and **SERVER ADMIN LOGIN NAME** to a notepad.

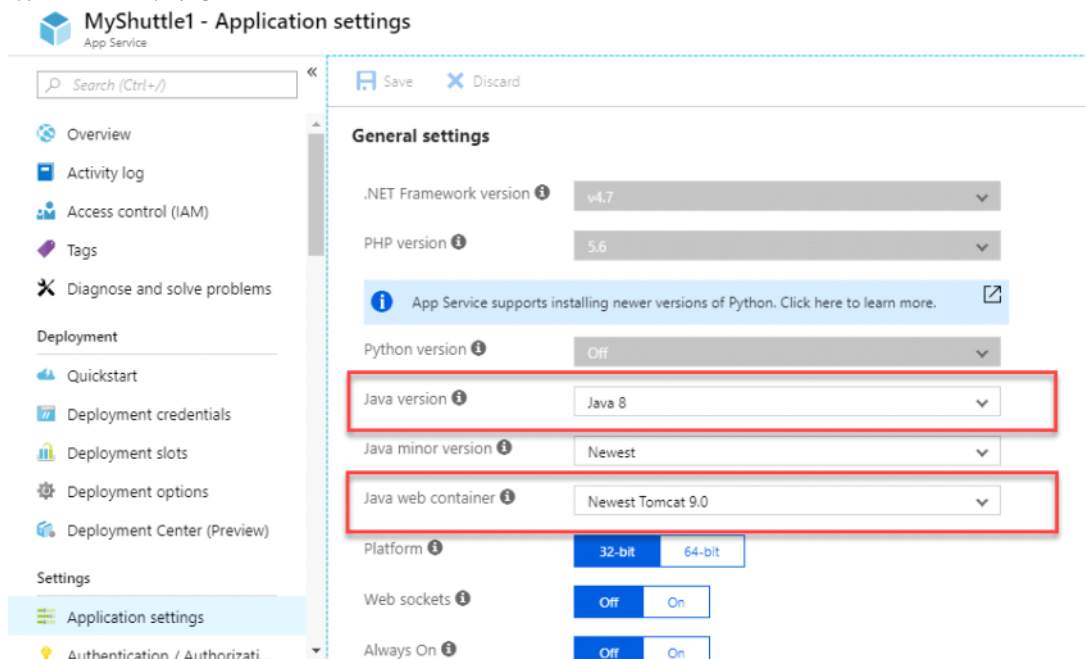


In this example, the server name is `myshuttle-1-mysqldbserver.mysql.database.azure.com` and the admin user name is `mysqlbuser@myshuttle-1-mysqldbserver`.

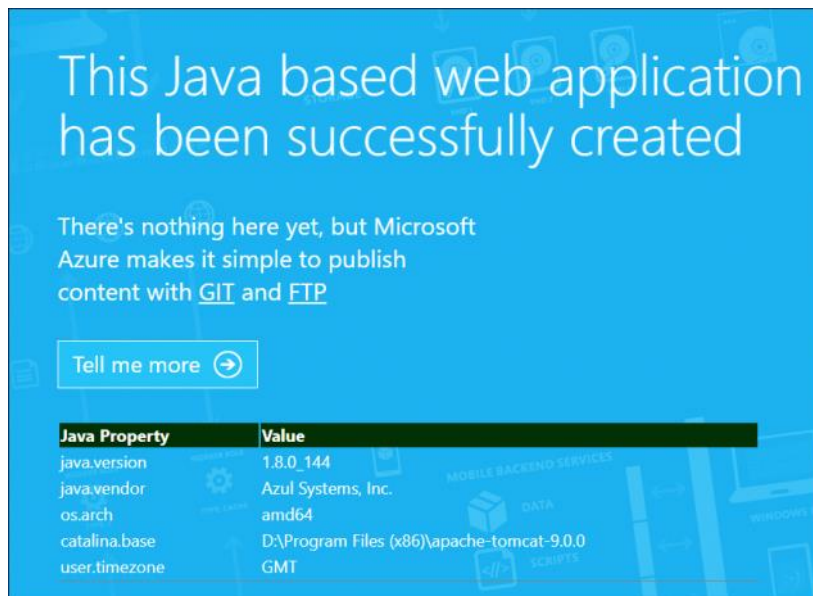
Updating the App Settings for the Web App

Next, navigate to the Web app that you have created. As you are deploying a Java application, you need to change the web app's web container to Apache Tomcat.

1. Click **Application Settings**. To change it to Tomcat, you will first need to install Java. Select a **Java Version** to install and then change **Web container** to use Apache Tomcat. For this purpose of the lab, you will choose **Java 8** and **Apache Tomcat 9.0** though the version number would not matter much for the simple app that we are deploying.

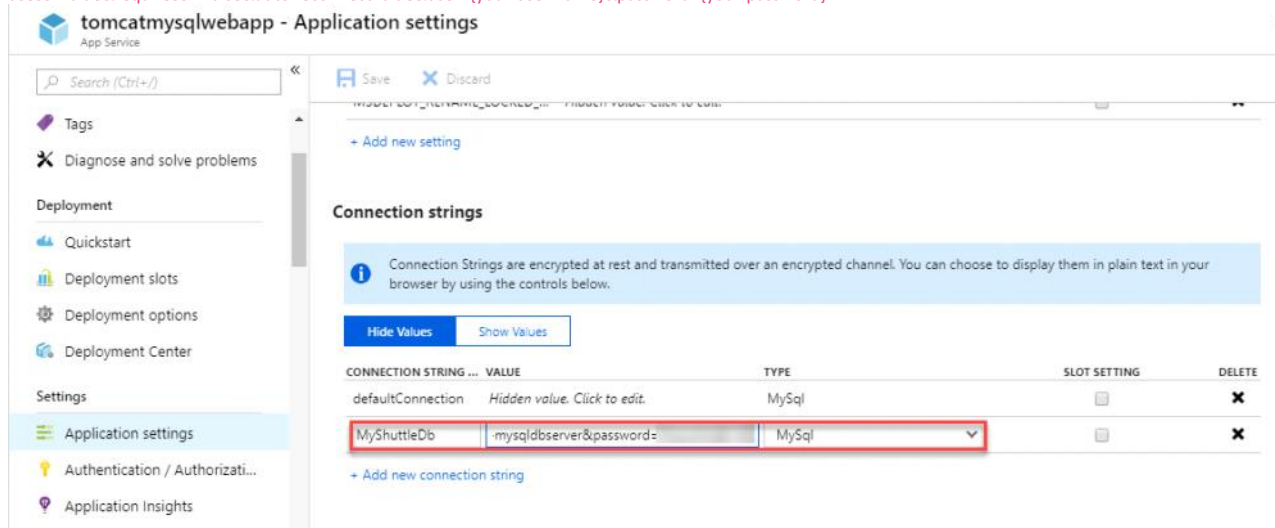


2. Click on **Save** and wait for the update to be applied. The web page will now look like the below image.



Next, you need to update the connection strings for the web app to connect to the database correctly. There are multiple ways you can do this - but for the purpose of this lab, you will take a simple approach by updating it directly on the Azure portal.

- From the Azure portal, select the Web app you provisioned. Select **Application Settings** and scroll down to the **Connection Strings** section.
- Add a new **MySQL** connection string with **MyShuttleDb** as the name, paste the following string for the value and replace **MySQL Server Name**, **your user name** and **your password** with the appropriate values -
`jdbc:mysql://{MySQL Server Name}:3306/alm?useSSL=true&requireSSL=false&autoReconnect=true&user={your user name}&password={your password}`



- MySQL Server Name : Value that you copied previously from the MySQL server Properties.
- your user name : Value that you copied previously from the MySQL server Properties.
- your password : Value that you provided during the creation of MYSQL database server in the *Deploy to Azure* phase.

- Click on **Save** to save the connection string.

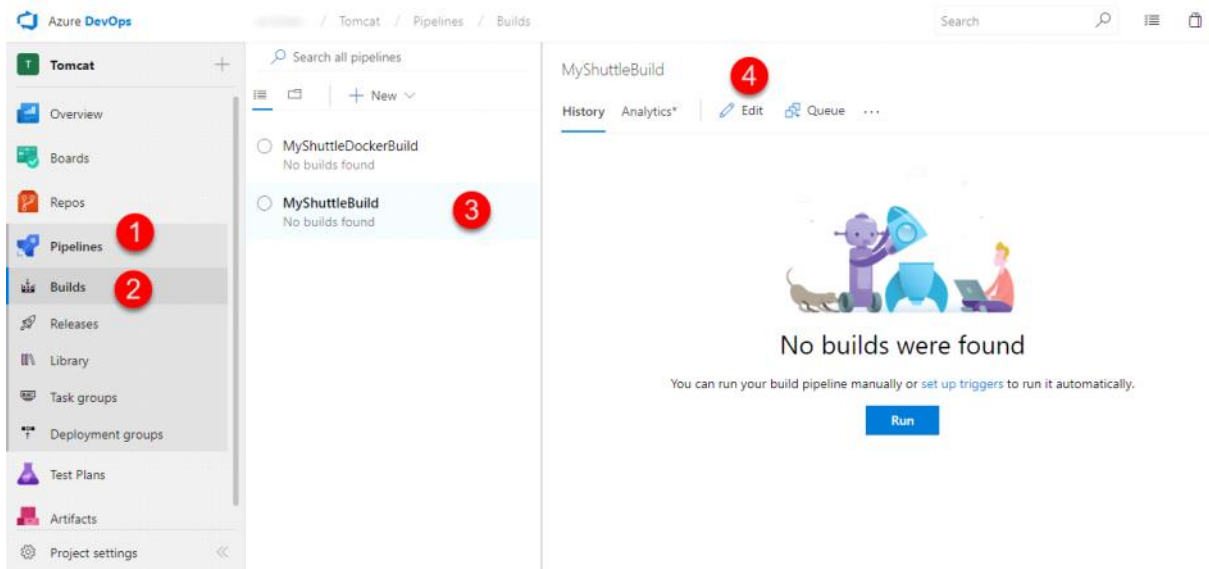
Note: Connection Strings configured here will be available as environment variables, prefixed with connection type for Java apps (also for PHP, Python and Node apps). In the **DataAccess.java** file under `src/main/java/com/microsoft/example` folder, we retrieve the connection string using the following code

```
String conStr = System.getenv("MYSQLCONNSTR_MyShuttleDb");
```

You have now setup and configured all the resources that is needed to deploy and run the MyShuttle application.

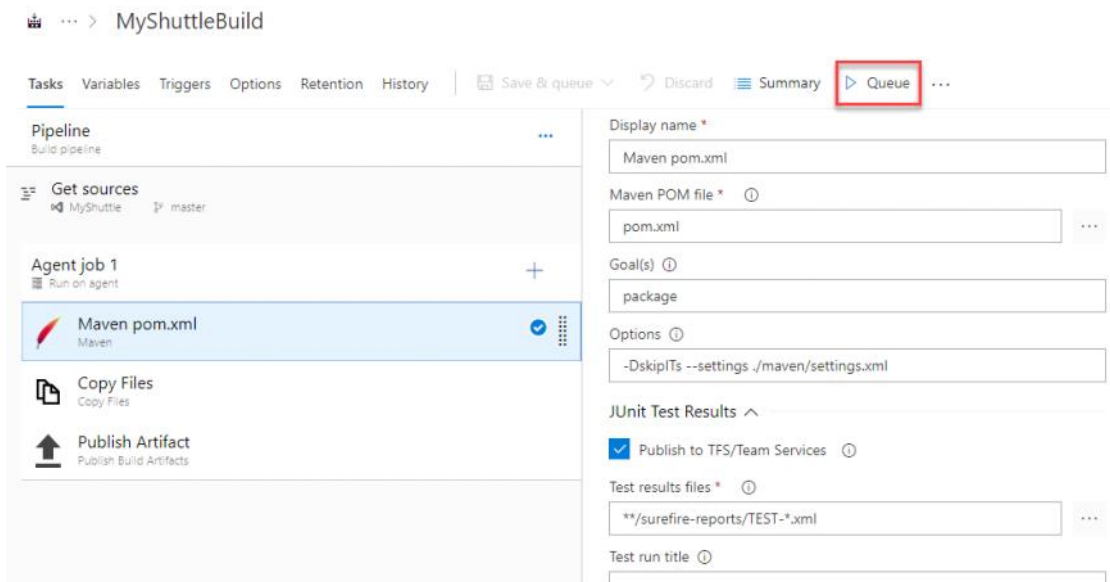
Deploy the changes to Web App

- Select **Pipelines** and then, **Builds**. Choose the build **MyShuttleBuild** and click **Edit Pipeline** to view the build definition.



The lab uses the standard **Maven** build template to compile the code, copy and publish the resulting artifacts for deployment. An additional file which is copied here is the *CreateMySQLDB.sql* file which creates a MySQL database and inserts a few records into it during the deployment.

- Click **Queue** to queue the build and wait for the build to complete.



Queue build for MyShuttleBuild

Agent pool
 Hosted VS2017

Branch
 master

Commit

Variables Demands

system.debug false

+ Add

Queue Cancel

- Once the build succeeds, Select **Releases** under **Pipelines**.
- Select **MyShuttle Release** and click **Edit Pipeline** to open the release definition.

Azure DevOps / MyShuttle / Pipelines / Releases

MyShuttle

- Overview
- Boards
- Repos
- Pipelines
- Builds
- Releases**

Release pipelines

Active All pipelines + New Edit pipeline Create a release ...

Search

No favorites yet

Recent

MyShuttle Release
 No deployments yet

- Make sure the artifact is pointing to the **Build** artifact as shown below. If you are following this lab from Jenkins hands-on-lab, make sure the artifact is pointing to Jenkins.

All pipelines > MyShuttle Release

Pipeline Tasks Variables Retention Options History

Artifacts + Add

MyShuttle Build

Stages + Add

Azure-Dev
 1 job, 1 task

Artifact

Build - MyShuttle Build

Project *
 MyShuttle

Source (build pipeline) *
 MyShuttleBuild

Default version *
 Latest

Source alias
 MyShuttle Build

- Click **Tasks**, select **Azure-Dev** and choose the **Azure subscription** details from the drop down.

Click **Authorize** and login to your Azure subscription in the pop-up window. Provide or choose the created **App Service Name** with the web app that you created previously in this lab.

7. Click **Tasks** and select **Execute Azure MySQL : SqlTaskFile** task and provide the following details.

- Azure Subscription Details : Select the appropriate subscription.
- Host Name : Select the **MySQL Database server** host name that was created.
- Server Admin Login : Provide the **SERVER ADMIN LOGIN NAME** that you noted down previously.
- Password : Provide the password that you created during the creation of *Azure Web App + MYSQL* database server in the Azure portal.

8. Select the **Deploy Azure App Service** task and ensure that the created **App service name** is reflected correctly.

Note: We are using the **Deploy Azure App Service** task. This task is used to update Azure App Service to deploy Web Apps and WebJobs to Azure. The task works on cross platform agents running Windows, Linux or Mac and uses the underlying deployment technologies of Web Deploy and Kudu. The task works for ASP.NET, ASP.NET Core 1 and Node.js based web applications. Note that this task works with Azure Resource Manager APIs only.

9. Click on **Save** and then **+Release | Create Release** to start a new release

All pipelines > MyShuttleRelease

Pipeline Tasks Variables Retention Options History

Azure-Dev
Deployment process

Run on agent
Run on agent

Execute Azure MySQL : SqlTaskFile
PREVIEW Azure Database for MySQL Deployment

Deploy Azure App Service
Azure App Service Deploy

Save + Release ...

+ Create a release
+ Create a draft release

Azure App Service Deploy

Version 3.*

Display name *
Deploy Azure App Service

Azure subscription * | Manage

Visual Studio Enterprise

Scoped to subscription 'Visual Studio Enterprise - MPN'

App name *

10. Wait for the release to complete. Then navigate to the Web App and select the **URL** from the overview blade. Add **/myshuttledev** context to the URL. For instance - <http://myshuttle1.azurewebsites.net/myshuttledev>
11. Select **Login** and try logging in to the site with any one of the following credentials.

Username	Password
barney	barneypassword
fred	fredpassword