



Microsoft Cloud Workshop

Cloud Native Applications

Before the hands-on lab setup guide

February 2020

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links may be provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

© 2020 Microsoft Corporation. All rights reserved.

Contents

- [Cloud-native applications before the hands-on lab setup guide](#)
 - [Requirements](#)
 - [Before the hands-on lab](#)
 - [Task 1: Setup Azure Cloud Shell](#)
 - [Task 2: Download Starter Files](#)
 - [Task 3: Resource Group](#)
 - [Task 4: Create an SSH key](#)
 - [Task 5: Create a Service Principal](#)
 - [Task 6: Deploy ARM Template](#)
 - [Task 7: Setup Azure DevOps project](#)
 - [Task 8: Connect securely to the build agent](#)
 - [Task 9: Complete the build agent setup](#)
 - [Task 10: Clone Repositories to the Build Agent](#)

Cloud-native applications before the hands-on lab setup guide

Requirements

1. Microsoft Azure subscription must be pay-as-you-go or MSDN.
 - Trial subscriptions will *not* work.
 - To complete this lab setup (including [Task 5: Create a Service Principal](#)) ensure your account includes the following:
 - Has the [Owner](#) built-in role for the subscription you use.
 - Is a [Member](#) user in the Azure AD tenant you use. (Guest users will not have the necessary permissions).
 - Note** If you do not meet these requirements, ask another member user with subscription owner rights to login to the portal and execute the task to create the service principal.
 - You must have enough cores available in your subscription to create the build agent and Azure Kubernetes Service cluster in [Task 6: Deploy ARM Template](#). You'll need eight cores if following the exact instructions in the lab, more if you choose additional agents or larger VM sizes. Execute the steps required before the lab to see if you need to request more cores in your sub.
2. An account in Azure DevOps.
3. Local machine or a virtual machine configured with:

- A browser, preferably Chrome for consistency with the lab implementation tests.
4. You will be asked to install other tools throughout the exercises.

Before the hands-on lab

Duration: 1 hour

You should follow all of the steps provided in this section *before* taking part in the hands-on lab ahead of time as some of these steps take time.

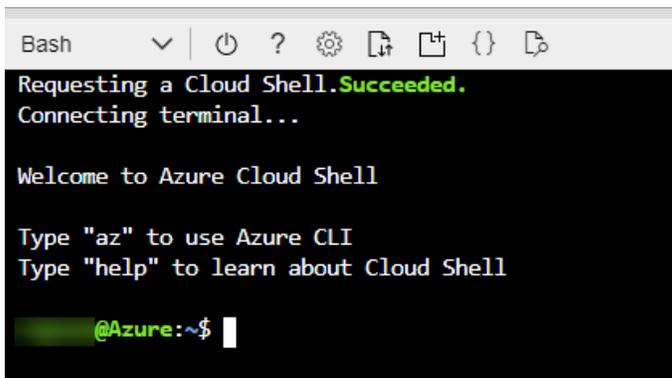
Task 1: Setup Azure Cloud Shell

1. Open a cloud shell by selecting the cloud shell icon in the menu bar.



The cloud shell icon is highlighted on the menu bar.

2. The cloud shell opens in the browser window. Choose “Bash” if prompted or use the left-hand dropdown on the shell menu bar to choose “Bash” (as shown).



This is a screenshot of the cloud shell opened in a browser window. Bash was selected.

3. You should make sure to set your default subscription correctly. To view your current subscription type:

```
az account show
```



```
Bash
:~$ az account list
[
  {
    "cloudName": "AzureCloud",
    "id": " ",
    "isDefault": false,
    "name": " ",
    "state": "Enabled",
    "tenantId": " ",
    "user": {
      "name": " ",
      "type": "user"
    }
  },
  {
    "cloudName": "AzureCloud",
    "id": " ",
    "isDefault": true,
    "name": " ",
    "state": "Enabled",
    "tenantId": " ",
    "user": {
      "name": " ",
      "type": "user"
    }
  }
]
:~$
```

In this screenshot of a Bash window, az account list has been typed and run at the command prompt. Some subscription information is visible in the window, and some information is obscured.

5. To set your default subscription to something other than the current selection, type the following, replacing {id} with the desired subscription id value:

```
az account set --subscription {id}
```

Task 2: Download Starter Files

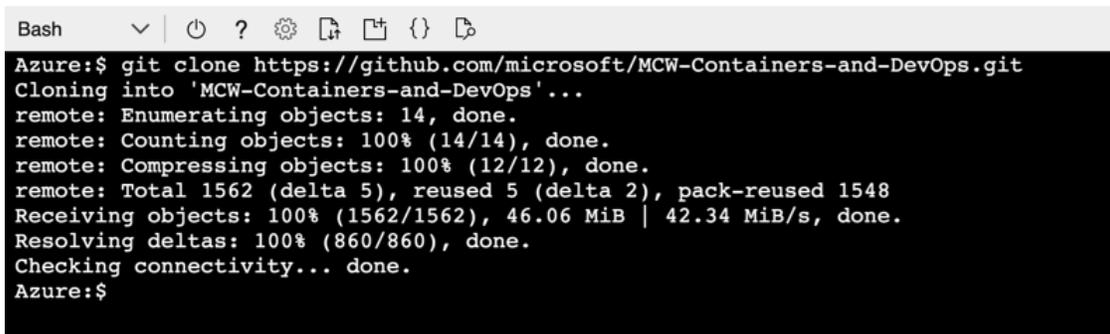
In this task, you use git to copy the lab content to your cloud shell so that the lab starter files will be available.

Note: If you don't have a cloud shell available, refer back to [Task 1: Setup Azure Cloud Shell](#).

1. Type the following command and press <ENTER>:

```
git clone https://github.com/microsoft/MCW-Cloud-native-applications.git
```

2. The lab files download.

A screenshot of a Bash terminal window. The terminal shows the command 'git clone https://github.com/microsoft/MCW-Containers-and-DevOps.git' being executed. The output indicates that the repository is being cloned into a folder named 'MCW-Containers-and-DevOps'. The process includes enumerating 14 objects, counting 100% of them, compressing 100% of them, and receiving 100% of 1562 objects (46.06 MiB) at a rate of 42.34 MiB/s. The process also shows resolving deltas (100% of 860) and checking connectivity, all of which are completed successfully. The prompt returns to 'Azure:\$'.

In this screenshot of a Bash window, git clone has been typed and run at the command prompt. The output from git clone is shown.

3. We do not need the .git folder, and later steps will be less complex if we remove it. Run this command:

```
rm -rf MCW-Cloud-native-applications/.git
```

Task 3: Resource Group

Create an Azure Resource Group to hold most of the resources that you create in this hands-on lab. This approach makes it easier to clean up later.

1. In your cloud shell window, you type a command similar to the following command:

Note: If you don't have a cloud shell available, refer back to [Task 1: Setup Azure Cloud Shell](#).

```
az group create -l [LOCATION] -n fabmedical-[SUFFIX]
```

- **Suffix:** Throughout the lab, suffix should be used to make resources unique, like your email prefix or your first initial and last name.
- **Location:** Choose a region where all Azure Container Registry SKUs have to be available, which is currently: Canada Central, Canada East, North Central US, Central US, South Central US, East US, East US 2, West US, West US 2, West Central US, France Central, UK South, UK West, North Europe, West Europe, Australia East, Australia Southeast, Brazil South, Central India, South India, Japan East, Japan West, Korea Central, Southeast Asia, East Asia, and remember this for future steps so that the resources you create in Azure are all kept within the same region.

Example:

```
az group create -l westus -n fabmedical-sol
```

2. When this completes, the Azure Portal shows your Resource Group.

Dashboard > Resource groups

Resource groups

vraposo

+ Add | ≡ Edit columns | ↻ Refresh | 🏷 Assign tags | ↓ Export to CSV

Subscriptions: 2 of 3 selected – Don't see a subscription? [Open Directory + Subscription settings](#)

fabmedical | 2 subscriptions | All local

1 items

<input type="checkbox"/>	NAME ↑↓
<input type="checkbox"/>	 fabmedical-

In this screenshot of the Azure Portal, the fabmedical-sol Resource group is listed.

Task 4: Create an SSH key

You create VMs during the upcoming exercises. In this section, you create an SSH key to access the VMs securely.

1. From the cloud shell command line, enter the following command to ensure that a directory for the SSH keys exists. You can ignore any errors you see in the output.

Note: If you don't have a cloud shell available, refer back to [Task 1: Setup Azure Cloud Shell](#).

```
mkdir .ssh
```

2. From the cloud shell command line, enter the following command to generate an SSH key pair. You can replace "admin" with your preferred name or handle.

```
ssh-keygen -t RSA -b 2048 -C admin@fabmedical
```

3. When asked to save the generated key to a file, enter .ssh/fabmedical for the name.
4. Enter a passphrase when prompted, and **don't forget it!**
5. Because you entered ".ssh/fabmedical", ssh-keygen generates the file in the ".ssh" folder in your user folder, where the cloud shell opens by default.

```
Bash
@Azure:~$ ssh-keygen -t RSA -b 2048 -C admin@fabmedical
Generating public/private RSA key pair.
Enter file in which to save the key (/home/ / .ssh/id_rsa): .ssh/fabmedical
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in .ssh/fabmedical.
Your public key has been saved in .ssh/fabmedical.pub.
The key fingerprint is:
SHA256:R7J+ufHw2zuPGe8KEdwfo4r1lMZ/exkRd/KCqNguiDk admin@fabmedical
The key's randomart image is:
+---[RSA 2048]---+
|
| . . . . o |
| . . . o . = + |
| + . o . + o + |
| o S . . . * . o |
| . . . o . = . . |
| o . . . = . . . + |
| E . . . * o . * + |
| . . . + . * B = |
+---[SHA256]---+
@Azure:~$
```

In this screenshot of the cloud shell window, `ssh-keygen -t RSA -b 2048 -C admin@fabmedical` has been typed and run at the command prompt. Information about the generated key appears in the window.

6. From the cloud shell command line, enter the following command to output the public key content. Copy this information to use later.

```
cat .ssh/fabmedical.pub
```

7. Keep this cloud shell open and remain in the default directory. You will use this shell in later tasks.

```
Bash
@Azure:~$ cat .ssh/fabmedical.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDUB2N7y68UTg6TbkD9uTd1K3BIQh3Xwa+TYieu3MI2I/1cN01I8+DzSoUA0
LkoVFNZsbsfYbWw
EVmJBOcHyeFBOP9
ZQYEYTD admin@fab
abmedical
@Azure:~$
```

In this screenshot of the cloud shell window, `cat .ssh/fabmedical` has been typed and run at the command prompt. Information about the public key content appears in the window.

Task 5: Create a Service Principal

Azure Kubernetes Service requires an Azure Active Directory service principal to interact with Azure APIs. The service principal is needed to dynamically manage resources such as user-

defined routes and the Layer 4 Azure Load Balancer. The easiest way to set up the service principal is by using the Azure cloud shell.

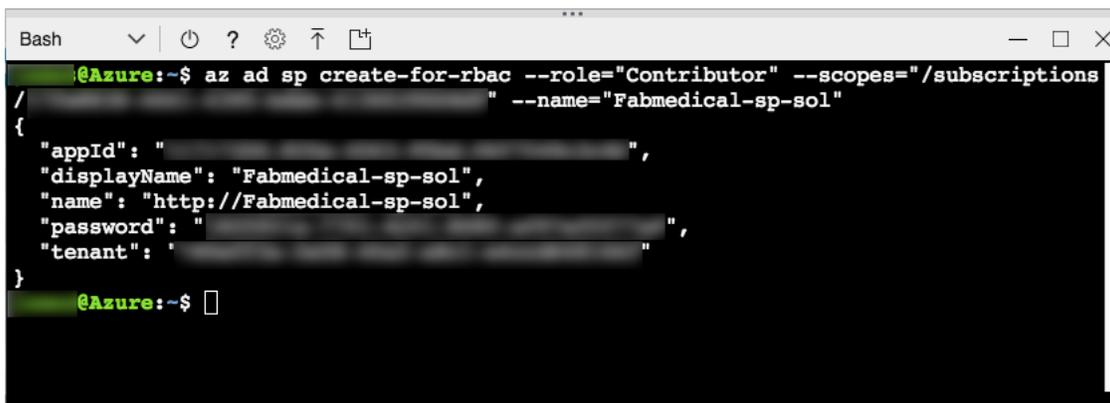
Note: To complete this task, ensure your account is an [Owner](#) built-in role for the subscription you use and is a [Member](#) user in the Azure AD tenant you use. You may have trouble creating a service principal if you do not meet these requirements.

1. To create a service principal, type the following command in the cloud shell command line, replacing {id} with your subscription identifier, and replacing suffix with your chosen suffix to make the name unique:

Note: If you don't have a cloud shell available, refer back to [Task 1: Setup Azure Cloud Shell](#).

```
az ad sp create-for-rbac --role="Contributor" --scopes="/subscriptions/{id}" --name="http://Fabmedical-sp-{SUFFIX}"
```

2. The command produces output like this. Copy this information to use later.



```
Bash
@Azure:~$ az ad sp create-for-rbac --role="Contributor" --scopes="/subscriptions
/
" --name="Fabmedical-sp-sol"
{
  "appId": " ",
  "displayName": "Fabmedical-sp-sol",
  "name": "http://Fabmedical-sp-sol",
  "password": " ",
  "tenant": " "
}
@Azure:~$
```

In this screenshot of a Bash window, `az ad sp create-for-rbac --role="Contributor" --scopes="/subscriptions/{id}" --name="Fabmedical-sp-SUFFIX"` has been typed and run at the command prompt. Service principal information is visible in the window, but at this time, we are unable to capture all of the information in the window. Future versions of this course should address this.

3. To get the service principal object id, type the following command, replacing {appId} with your service principal appId:

```
az ad sp show --id {appId} --query "{objectId:@.objectId}"
```

4. The command produces output like this. Copy this information to use later.

```
Bash
@Azure:~$ az ad sp show --id d41261a3-d8b8-4cf0-890d-1fb6efc20a67 --query "{objectId:@.objectId}"
{
  "objectId": "d41261a3-d8b8-4cf0-890d-1fb6efc20a67"
}
@Azure:~$
```

In this screenshot of a Bash window, `az ad sp show --id d41261a3-d8b8-4cf0-890d-1fb6efc20a67 --query "{objectId:@.objectId}"` has been typed and run at the command prompt. Service Principal information is visible in the window.

Task 6: Deploy ARM Template

In this section, you configure and execute an ARM template that creates all the resources that you need throughout the exercises.

1. In Azure cloud shell, switch to the ARM template directory:

Note: If you don't have a cloud shell available, refer back to [Task 1: Setup Azure Cloud Shell](#).

```
cd MCW-Cloud-native-applications/Hands-on\ lab/arm/
```

2. Open the `azuredeploy.parameters.json` file for editing using Azure Cloud Shell editor.

code `azuredeploy.parameters.json`

```
Bash
1
2 "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
3 "contentVersion": "1.0.0.0",
4 "parameters": {
5   "Suffix": {
6     "value": "SUP"
7   },
8   "VirtualMachineAdminUsernameLinux": {
9     "value": "adminfabmedical"
10  },
11  "VirtualMachineAdminPublicKeyLinux": {
12    "value": "ssh-rsa AAAAB3N(...)vPiybQV admin@fabmedical"
13  },
14  "KubernetesServicePrincipalClientId": {
15    "value": "GUID"
16  },
17  "KubernetesServicePrincipalClientSecret": {
18    "value": "GUID"
19  },
20  "KubernetesServicePrincipalObjectId": {
21    "value": "GUID"
22  },
23  "CosmosLocation": {
24    "value": "location"
25  },
26  "CosmosLocationName": {
27    "value": "Location Name"
28  },
29  "CosmosPairedLocation": {
30    "value": "pairedlocation"
31  },
32  "CosmosPairedLocationName": {
33    "value": "Paired Location Name"
34  }
35 }
36
```

@Azure:~/MCW-Containers-and-DevOps/Hands-on lab/arm\$ code azuredeploy.parameters.json
@Azure:~/MCW-Containers-and-DevOps/Hands-on lab/arm\$

This screenshot shows the online editor for azure cloud shell.

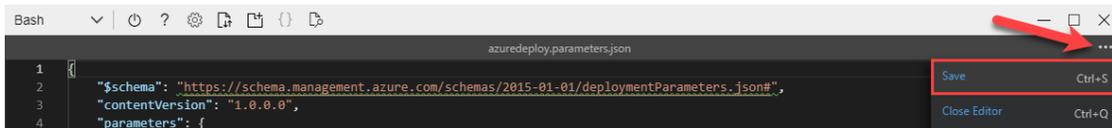
3. Update the values for the various keys so that they match your environment:

- **Suffix:** Enter a shortened version of your SUFFIX with a max of 3 chars.
- **VirtualMachineAdminUsernameLinux:** The Linux Build Agent VM admin username (example: "adminfabmedical").
- **VirtualMachineAdminPublicKeyLinux:** The Linux Build Agent VM admin ssh public key. You find this value in the .ssh/fabmedical.pub file created previously (example: "ssh-rsa AAAAB3N(...)vPiybQV admin@fabmedical").
- **KubernetesServicePrincipalClientId:** The Kubernetes Cluster Service Principal Client Id. Use the service principal "appld" from a previous step.
- **KubernetesServicePrincipalClientSecret:** The Kubernetes Cluster Service Principal Client Secret. Use the service principal "password" from a previous step.
- **KubernetesServicePrincipalObjectId:** The Kubernetes Cluster Service Principal Object Id. Use the service principal "objectId" from a previous step.
- **CosmosLocation:** The primary location of the Azure Cosmos DB. Use the same location as the resource group previously created (example: "eastus").
- **CosmosLocationName:** The name of the primary location of the Azure Cosmos DB. Use the name of the same location as the resource group previously created (example: "East US").
- **CosmosPairedLocation:** The secondary location of the Azure Cosmos DB. Use a location from the list below (example: "westus").
- **CosmosPairedLocationName:** The name of the secondary location of the Azure Cosmos DB. Use the location name from the list below that matches the secondary location defined in the previous key (example: "West US").

Location	Location Name
canadacentral	Canada Central
canadaeast	Canada East
northcentralus	North Central US
centralus	Central US
southcentralus	South Central US
eastus	East US
eastus2	East US 2
westus	West US
westus2	West US 2
westcentralus	West Central US
francecentral	France Central
uksouth	UK South
ukwest	UK West
northeurope	North Europe
westeurope	West Europe

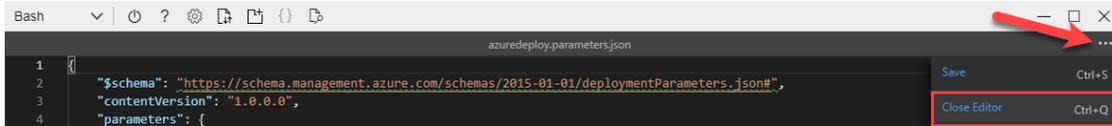
australiaeast	Australia East
australiasoutheast	Australia Southeast
brazilsouth	Brazil South
centralindia	Central India
southindia	South India
japaneast	Japan East
japanwest	Japan West
koreacentral	Korea Central
southeastasia	Southeast Asia
eastasia	East Asia

4. Select the ... button and select **Save**.



In this screenshot of an Azure Cloud Shell editor window, the ... button has been selected, and the Save option is highlighted.

5. Select the ... button again and select **Close Editor**.



In this screenshot of the Azure Cloud Shell editor window, the ... button has been selected, and the Close Editor option is highlighted.

6. Create the needed resources by typing the following instruction (case sensitive), replacing {resourceGroup} with the name of the previously created resource group:

```
az deployment group create --resource-group {resourceGroup} --template-file azuredeploy.json --parameters azuredeploy.parameters.json
```

This command takes up to 30 to 60 minutes to deploy all lab resources. You can continue to the next task to setup Azure DevOps while the deployment runs.

Task 7: Setup Azure DevOps project

FabMedical has provided starter files for you. They have taken a copy of the websites for their customer Contoso Neuro and refactored it from a single node.js site into a website with a content API that serves up the speakers and sessions. This refactored code is a starting point to validate the containerization of their websites. Use this to help them complete a POC that validates the development workflow for running the website and API as Docker containers and managing them within the Azure Kubernetes Service environment.

1. Open a **new** Azure Cloud Shell console.
2. Navigate to the FabMedical source code folder and list the contents.

```
cd ~/MCW-Cloud-native-applications/Hands-on\ lab/lab-files/developer/  
ll
```

Important note: If you will be taking the Infrastructure edition of the lab, instead of using the above instructions, type the following ones:

```
cd ~/MCW-Cloud-native-applications/Hands-on\ lab/lab-files/infrastructure  
/  
ll
```

This will take you to the version of the starter files that will be used by that edition of the lab.

3. You'll see the listing includes three folders, one for the web site, another for the content API and one to initialize API data:

```
content-api/  
content-init/  
content-web/
```

4. Set your username and email, which git uses for commits.

```
git config --global user.email "you@example.com"  
git config --global user.name "Your Name"
```

5. Configure git CLI to cache your credentials, so that you don't have to keep re-typing them.

```
git config --global credential.helper cache
```

6. Open a new browser tab to visit [Azure DevOps](#) and log into your account.

If you have never logged into this account, Azure DevOps takes you through a first-run experience:

- Confirm your contact information and select next.
- Select "Create new account".
- Enter a fabmedical-SUFFIX for your account name and select Continue.

7. Create an Azure DevOps Project.

- Enter fabmedical as the project name.
- Ensure the project is Private.
- Choose the "Advanced" dropdown.
- Ensure the Version control is set to Git.
- Select the "Create" button.

Create new project



Project name *

fabmedical ✓

Description

Visibility

<input type="radio"/> Public ⓘ Anyone on the internet can view the project. Certain features like TFVC are not supported.	<input checked="" type="radio"/> Private Only people you give access to will be able to view this project.
---	--

Public projects are disabled for your organization. You can turn on public visibility with [organization policies](#).

^ Advanced

Version control ⓘ

Git

Work item process ⓘ

Agile

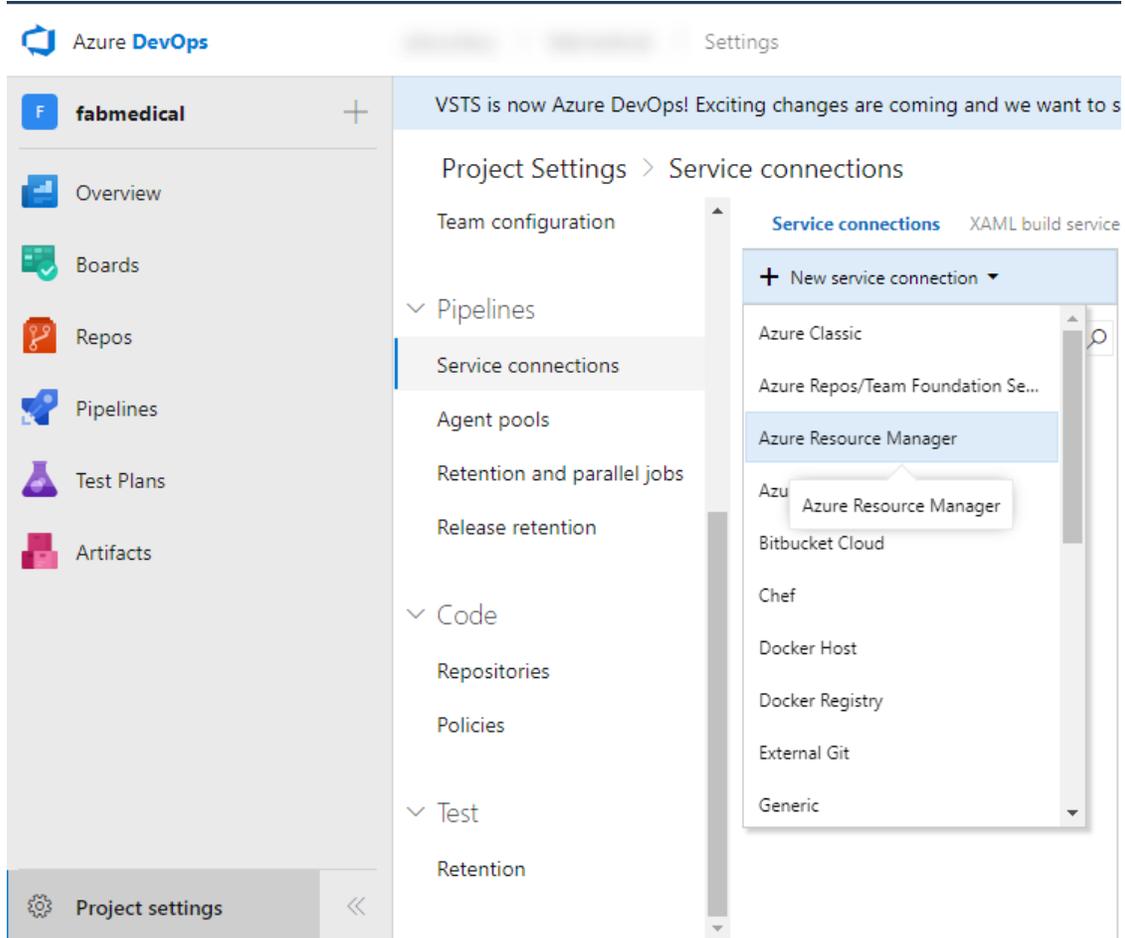


Cancel

Create

Create Project Dialog with an arrow pointing at the Create button

8. Enable multi-stage pipelines:
 - Select your user icon in the top right corner.
 - Then choose the three dots to access the “Preview Features” menu item.
 - Toggle multi-stage pipelines to “On”.
9. Next, add an Azure Service Connection to your Azure DevOps account. Select the Project settings gear icon to access your settings. Then select Service Connections.
10. Choose “+ New service connection”. Then pick “Azure Resource Manager” from the menu.



A screenshot of the New service connection selection in Azure DevOps with Azure Resource Manager highlighted.

11. Select the link indicated in the screenshot below to access the advanced settings.

Add an Azure Resource Manager service connection

Service Principal Authentication Managed Identity Authentication Publish Profile Based Authentication

Connection name

Scope level

Subscription

Resource Group

Subscriptions listed are from Azure Cloud

A new Azure service principal will be created and assigned with "Contributor" role, having access to all resources within the subscription. Optionally, you can select the Resource Group to which you want to limit access.

If your subscription is not listed above, or your organization is not backed by Azure Active Directory, or to specify an existing service principal, [use the full version of the service connection dialog.](#)

Allow all pipelines to use this connection.

OK

Close

A screenshot of the Add Azure Resource Manager dialog where you can enter your subscription information.

12. Enter the required information using the service principal information you created earlier.
 - **Connection name:** azurecloud
 - **Environment:** AzureCloud
 - **Scope Level:** Subscription
 - **Subscription ID:** Enter id from az account show output.
 - **Subscription name:** Enter name from az account show output.
 - **Service principal client ID:** Enter appld from service principal output.
 - **Service principal key:** Enter password from service principal output.
 - **Tenant ID:** Enter tenant from service principal output.

Add an Azure Resource Manager service connection ×

Service Principal Authentication Managed Identity Authentication Publish Profile Based Authentication

Connection name:

Environment:

Scope level:

Subscription ID:

Subscription name:

Service principal client ID:

Service principal key Certificate

Service principal key:

Tenant ID:

Connection: ✔ Verified Verify connection

For help on creating an Azure service principal, see [service connections](#).

To create a new service principal automatically, [use the automated version of the service connection dialog](#).

Allow all pipelines to use this connection.

A screenshot of the Add Resource Manager Add Service Endpoint dialog.

13. Select “Verify connection” then select “OK”.

Note: If the connection does not verify, then recheck and reenter the required data.

14. Next, add another Azure Service Connection to your Azure DevOps account. Select the Project settings gear icon to access your settings. Then choose Service Connections.
15. Choose “+ New service connection”. Then pick “Docker Registry” from the menu.

Add a Docker Registry service connection

Registry type * Docker Hub Others Azure Container Registry

Connection name *

Docker Registry ⓘ

Docker ID

Password

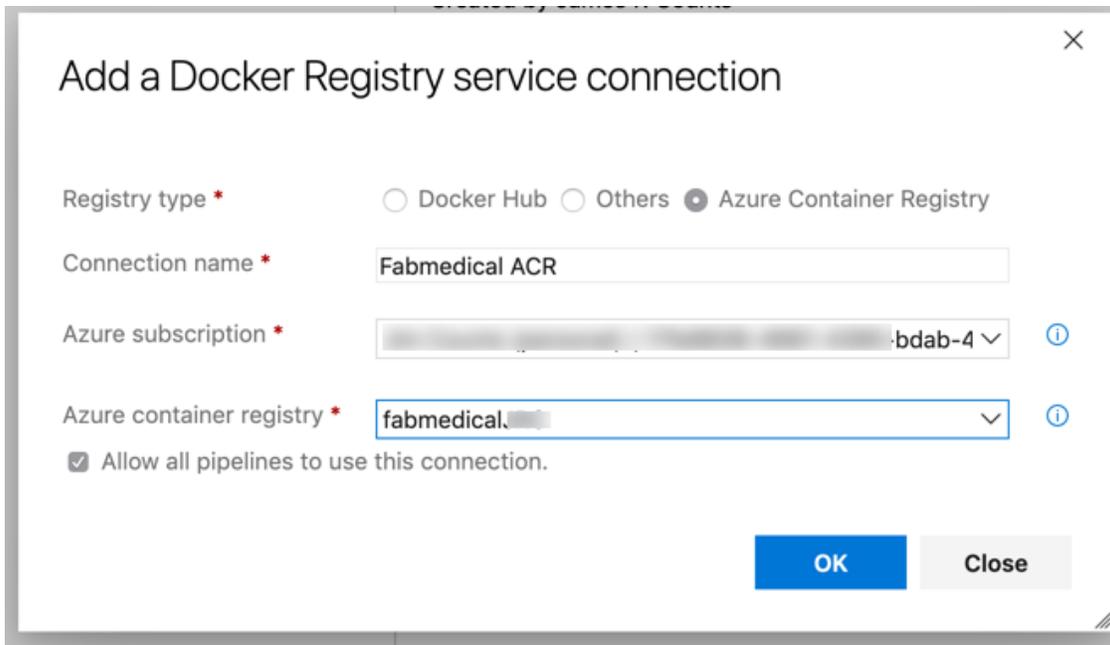
Email

Allow all pipelines to use this connection.

OK Close

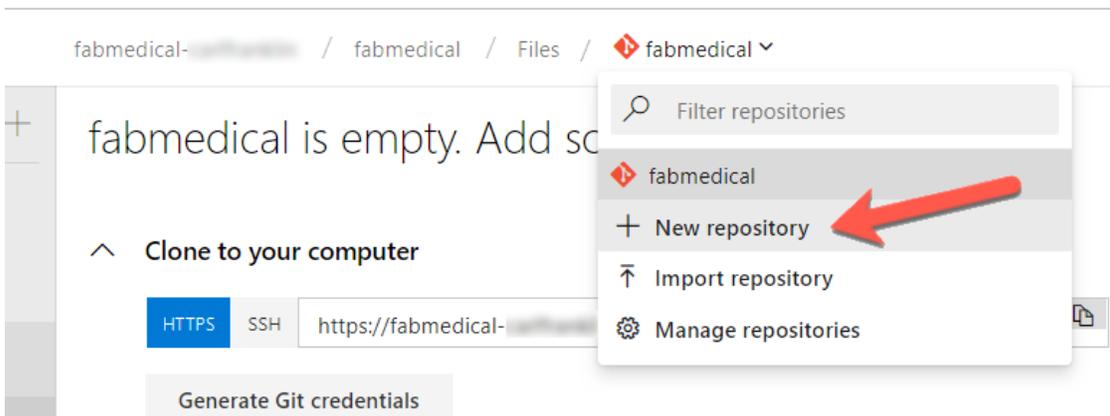
A screenshot of the Add Docker Registry Service Connection dialog.

16. Enter the required information using the service principal information you created earlier.
 - **Environment:** Azure Container Registry
 - **Connection name:** Fabmedical ACR
 - **Azure Subscription:** Choose the subscription you are using for the lab.
 - **Azure Container Registry:** Choose the registry created for you by the ARM deployment.



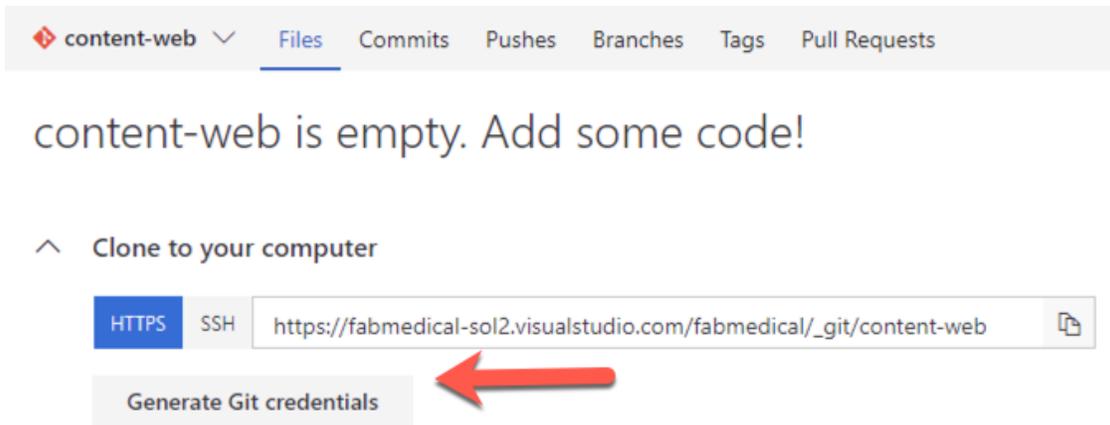
A screenshot of the Add Docker Registry Service Connection dialog with the values entered as described above.

17. Select "OK".
18. Next, choose "Repos" then use the repository dropdown to create a new repository by selecting "+ New repository".



The repository dropdown is displayed with the + New repository item selected.

- Enter "content-web" as the repository name.
- Once Azure DevOps creates the repository, select "Generate Git credentials".



The Clone to your computer section is displayed with the Generate Git Credentials button selected.

19. Copy the Personal Access Token and save it for later steps.
20. Using your cloud shell window, initialize a new git repository for content-web.

```
cd content-web
git init
git add .
git commit -m "Initial Commit"
```

21. Return to your Azure DevOps tab and copy the commands to add your Azure DevOps repository as a new remote for push. Copy the commands for “HTTPS” similar to this example:

```
git remote add origin https://fabmedical-sol@dev.azure.com/fabmedical-sol
/fabmedical/_git/content-web
git push -u origin --all
```

22. Now use the commands copied from Azure DevOps to configure the remote repository and push the code to Azure DevOps. When prompted for a password, paste your Azure DevOps Personal Access Token you copied earlier in this task.
23. Return to Azure DevOps and use the repository dropdown to create a second repository called content-api.

Note: You do not need to generate git credentials again. The same PAT works for both repositories.

24. Using your cloud shell window, initialize a new git repository in the content-api directory.

```
cd ../content-api
git init
git add .
git commit -m "Initial Commit"
```

25. Copy the commands to add your content-api repository as a new remote for push. Copy the commands for “HTTPS”.
26. Now use the commands copied from Azure DevOps to configure the remote repository and push the code to Azure DevOps. If prompted for a password, paste your Azure DevOps Personal Access Token you copied earlier in this task.
27. Use the repository drop down to create a third repository called content-init.

Note: You do not need to generate git credentials again. The same PAT works for both repositories.

28. Using your cloud shell window, initialize a new git repository in the content-init directory.

```
cd ../content-init
git init
git add .
git commit -m "Initial Commit"
```

29. Copy the commands to add your content-init repository as a new remote for push. Copy the commands for “HTTPS”.
30. Now use the commands copied from Azure DevOps to configure the remote repository and push the code to Azure DevOps. If prompted for a password, paste your Azure DevOps Personal Access Token you copied earlier in this task.

Task 8: Connect securely to the build agent

In this section, you validate that you can connect to the new build agent VM.

1. Open a **new** Azure Cloud Shell console and run the following command to find the IP address for the build agent VM provisioned when you ran the ARM deployment:

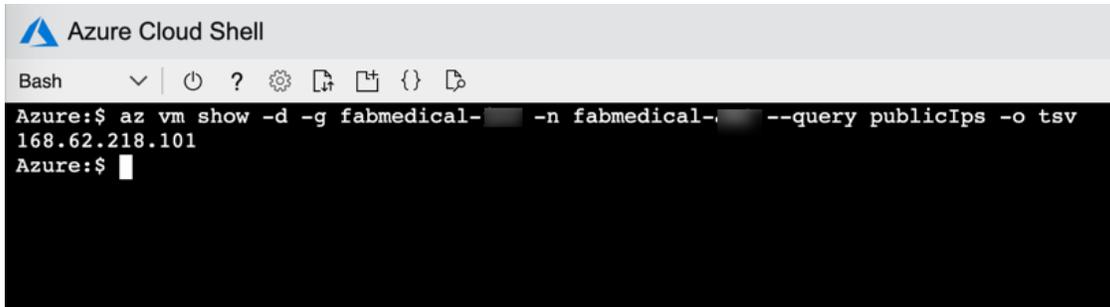
Note: If you don't have a cloud shell available, refer back to [Task 1: Setup Azure Cloud Shell](#).

```
az vm show -d -g fabmedical-[SUFFIX] -n fabmedical-[SHORT_SUFFIX] --query publicIps -o tsv
```

Example:

```
az vm show -d -g fabmedical-sol -n fabmedical-SOL --query publicIps -o ts
v
```

2. In the cloud shell output, take note of the public IP address for the VM.



```
Azure Cloud Shell
Bash
Azure:$ az vm show -d -g fabmedical- -n fabmedical- --query publicIps -o tsv
168.62.218.101
Azure:$
```

The cloud shell window is displayed with the Public IP address shown.

3. Connect to the new VM you created by typing the following command:

```
ssh -i [PRIVATEKEYNAME] [BUILDAGENTUSERNAME]@[BUILDAGENTIP]
```

Replace the bracketed values in the command as follows:

- [PRIVATEKEYNAME]: Use the private key name “.ssh/fabmedical,” created above.

```
ssh -i .ssh/fabmedical adminfabmedical@52.174.141.11
```

4. When asked to confirm if you want to connect, as the authenticity of the connection cannot be validated, type “yes”.
5. When asked for the passphrase for the private key you created previously, enter this value.
6. SSH connects to the VM and displays a command prompt such as the following. Keep this cloud shell window open for the next step:

```
adminfabmedical@fabmedical-SUFFIX:~$
```

```
Azure Cloud Shell
Bash
@Azure:~$ ssh -i .ssh/fabmedical adminfabmedical@23.97.213.98
The authenticity of host '23.97.213.98 (23.97.213.98)' can't be established.
ECDSA key fingerprint is SHA256:6ptcz3vihXphBPOqcZzc/qJEGdBm9VpfWGHMZN85Ck8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '23.97.213.98' (ECDSA) to the list of known hosts.
Enter passphrase for key '.ssh/fabmedical':
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-1063-azure x86_64)

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

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.

adminfabmedical@fabmedical-~$
```

In this screenshot of a Cloud Shell window, `ssh -i .ssh/fabmedical adminfabmedical@52.174.141.11` has been typed and run at the command prompt. The information detailed above appears in the window. At this time, we are unable to capture all of the information in the window. Future versions of this course should address this.

Note: If you have issues connecting, you may have pasted the SSH public key incorrectly in the ARM template. Unfortunately, if this is the case, you will have to recreate the VM and try again.

Task 9: Complete the build agent setup

In this task, you update the packages and install the Docker engine.

1. Go to the cloud shell window that has the SSH connection open to the build agent VM.
2. Update the Ubuntu packages and install curl and support for repositories over HTTPS in a single step by typing the following in a single line command. Respond by typing "Y" and pressing enter, if asked if you would like to proceed.

```
sudo apt-get update && sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

3. Add Docker's official GPG key by typing the following in a single line command:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

4. Add Docker's stable repository to Ubuntu packages list by typing the following in a single line command:

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

5. Add NodeJs PPA to use NodeJS LTS release and update the Ubuntu packages and install Docker engine, node.js, and the node package manager by typing the following commands, each on their own line. If asked if you would like to proceed, respond by typing "Y" and pressing enter.

```
sudo apt-get install curl python-software-properties
```

```
curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
```

```
sudo apt-get update && sudo apt-get install -y docker-ce nodejs mongodb-clients
```

6. Now, upgrade the Ubuntu packages to the latest version by typing the following in a single line command. If asked if you would like to proceed, respond by typing "Y" and pressing enter.

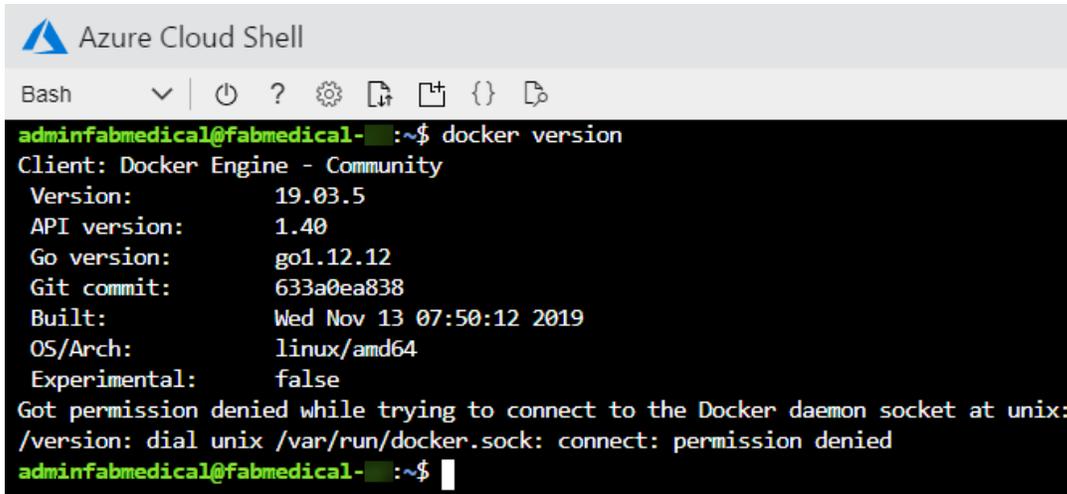
```
sudo apt-get upgrade
```

7. Install docker-compose

```
sudo curl -L https://github.com/docker/compose/releases/download/1.21.2/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose  
sudo chmod +x /usr/local/bin/docker-compose
```

8. When the command has completed, check the Docker version installed by executing this command. The output may look something like that shown in the following screenshot. Note that the server version is not shown yet, because you didn't run the command with elevated privileges (to be addressed shortly).

```
docker version
```



```
Azure Cloud Shell
Bash
adminfabmedical@fabmedical-:~$ docker version
Client: Docker Engine - Community
Version: 19.03.5
API version: 1.40
Go version: go1.12.12
Git commit: 633a0ea838
Built: Wed Nov 13 07:50:12 2019
OS/Arch: linux/amd64
Experimental: false
Got permission denied while trying to connect to the Docker daemon socket at unix:
/var/run/docker.sock: connect: permission denied
adminfabmedical@fabmedical-:~$
```

In this screenshot of a Cloud Shell window, docker version has been typed and run at the command prompt. Docker version information appears in the window.

9. You may check the versions of node.js and npm as well, just for information purposes, using these commands:

```
nodejs --version
```

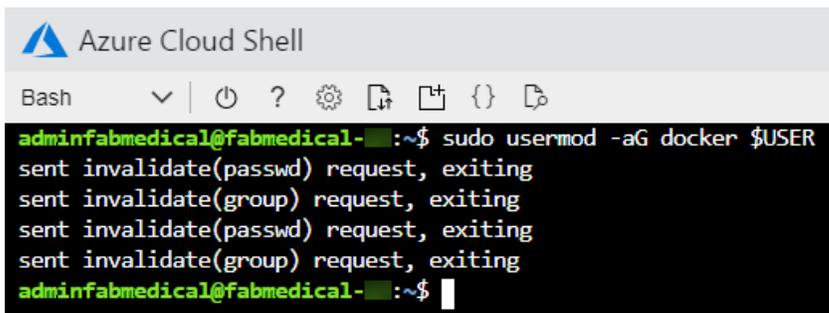
```
npm -version
```

10. Install the Angular CLI.

```
sudo npm install -g @angular/cli
```

11. To remove the requirement to use sudo, add your user to the Docker group. You can ignore any errors you see in the output.

```
sudo usermod -aG docker $USER
```

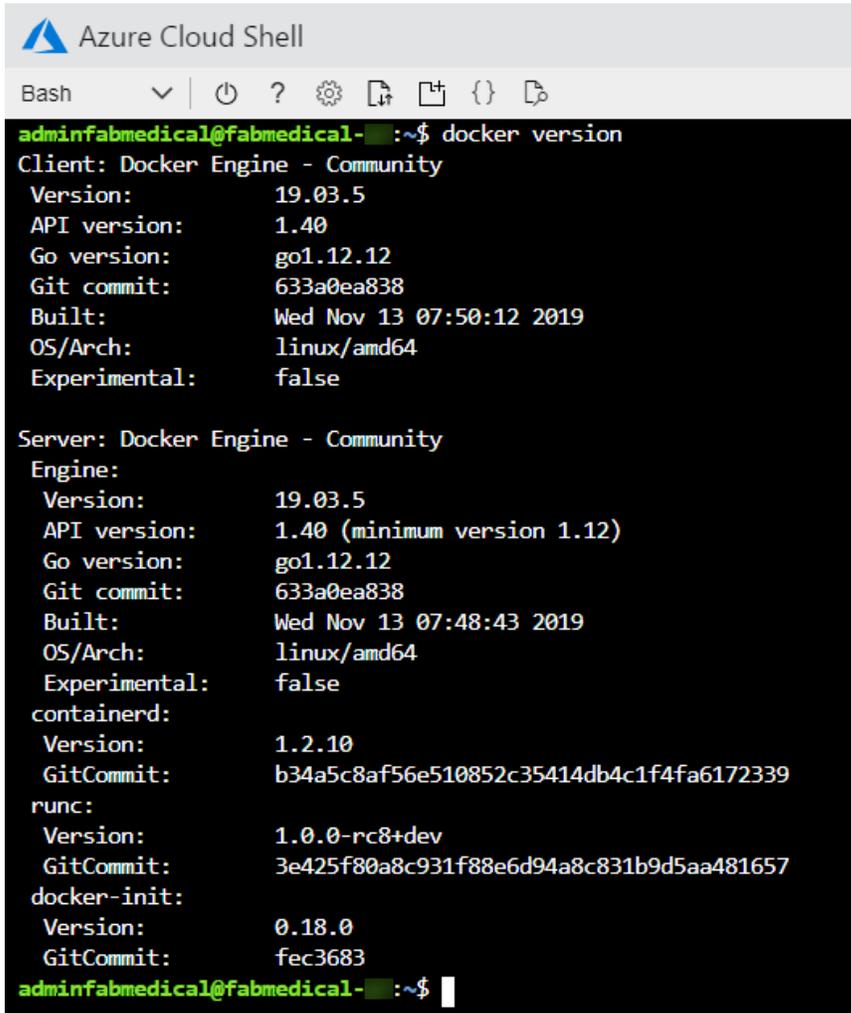


```
Azure Cloud Shell
Bash
adminfabmedical@fabmedical-:~$ sudo usermod -aG docker $USER
sent invalidate(passwd) request, exiting
sent invalidate(group) request, exiting
sent invalidate(passwd) request, exiting
sent invalidate(group) request, exiting
adminfabmedical@fabmedical-:~$
```

In this screenshot of a Cloud Shell window, sudo usermod -aG docker \$USER has been typed and run at the command prompt. Errors appear in the window.

12. For the user permission changes to take effect, exit the SSH session by typing 'exit', then press <Enter>. Reconnect to the build agent VM using SSH as you did in the previous task.

- Repeat the Docker version command, and note the output now shows the server version as well.



```
Azure Cloud Shell
Bash
adminfabmedical@fabmedical-:~$ docker version
Client: Docker Engine - Community
Version:      19.03.5
API version:  1.40
Go version:   go1.12.12
Git commit:   633a0ea838
Built:        Wed Nov 13 07:50:12 2019
OS/Arch:      linux/amd64
Experimental: false

Server: Docker Engine - Community
Engine:
Version:      19.03.5
API version:  1.40 (minimum version 1.12)
Go version:   go1.12.12
Git commit:   633a0ea838
Built:        Wed Nov 13 07:48:43 2019
OS/Arch:      linux/amd64
Experimental: false
containerd:
Version:      1.2.10
GitCommit:   b34a5c8af56e510852c35414db4c1f4fa6172339
runc:
Version:      1.0.0-rc8+dev
GitCommit:   3e425f80a8c931f88e6d94a8c831b9d5aa481657
docker-init:
Version:      0.18.0
GitCommit:   fec3683
adminfabmedical@fabmedical-:~$
```

In this screenshot of a Cloud Shell window, docker version has been typed and run at the command prompt. Docker version information appears in the window, in addition to server version information.

- Run a few Docker commands:

- One to see if there are any containers presently running.

```
docker container ls
```

- One to see if any containers exist, whether running or not.

```
docker container ls -a
```

- In both cases, you have an empty list but no errors while running the command. Your build agent is ready with the Docker engine running correctly.

```
Azure Cloud Shell
Bash
adminfabmedical@fabmedical-:~$ docker container ls
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
adminfabmedical@fabmedical-:~$ docker container ls -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
adminfabmedical@fabmedical-:~$
```

In this screenshot of a Cloud Shell window, docker container ls has been typed and run at the command prompt, as has the docker container ls -a command.

Task 10: Clone Repositories to the Build Agent

In this task, you clone your repositories from Azure DevOps so you can work with them on the build agent.

1. As you previously did in cloud shell, set your username and email which are used for git commits.

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

Note: In some cases, the root user owns your user's .config folder. If this happens, run the following command to return ownership to adminfabmedical and then try the git command again:

```
sudo chown -R $USER:$(id -gn $USER) /home/adminfabmedical/.config
```

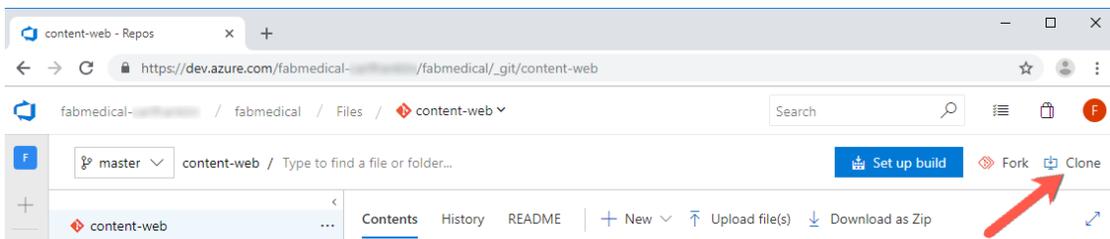
2. Configure git CLI to cache your credentials, so that you don't have to keep re-typing them.

```
git config --global credential.helper cache
```

Note: In some cases, the root user owns your user's .config folder. If this happens, run the following command to return ownership to adminfabmedical and then try the git command again:

```
sudo chown -R $USER:$(id -gn $USER) /home/adminfabmedical/.config
```

3. Visit the content-web repository in Azure DevOps and select "Clone" in the right corner.



The content-web repository page is displayed with the Clone button selected.

4. Copy the repository URL.
5. Use the repository URL to clone the content-web code to your build agent machine.

```
git clone <REPOSITORY_URL>
```

Note: In some cases, the root user owns your user's .config folder. If this happens, run the following command to return ownership to adminfabmedical and then try the git command again:

```
sudo chown -R $USER:$(id -gn $USER) /home/adminfabmedical/.config
```

6. When prompted for a password, use your PAT token from previous steps.
7. In your browser, switch to the content-api repository and select "Clone" to see and copy the repository URL.
8. Use the repository URL and git clone to copy the content-api code to your build agent.
9. In your browser, switch to the content-init repository and select "Clone" to see and copy the repository URL.
10. Use the repository URL and git clone to copy the content-init code to your build agent.

Note: Keep this cloud shell window open as your build agent SSH connection. The lab instructs you to open additional cloud shell sessions as and when needed.

You should follow all steps provided *before* performing the Hands-on lab.