JOURNEY TO THE CLOUD

A Developer's Perspective

DIONNE CONDOR-FARRELL

Senior Developer (Java) – Transport for London
Founding Member - UKBlackTech

Twitter: @dionnecf


MY JOURNEY TO THE CLOUD…
Frustration with limited resources

MY JOURNEY TO THE CLOUD…

Disparity between development teams

MY JOURNEY TO THE CLOUD…

Having to rely on another team

MY JOURNEY TO THE CLOUD…

Research Cloud Computing Providers

And many more…

BENEFITS OF EMBRACING THE CLOUD

- Saves Time
- Simplicity
- Flexibility
- Integration
- Save Your Sanity!

Microsoft Azure

Azure App Service

=

Web Apps

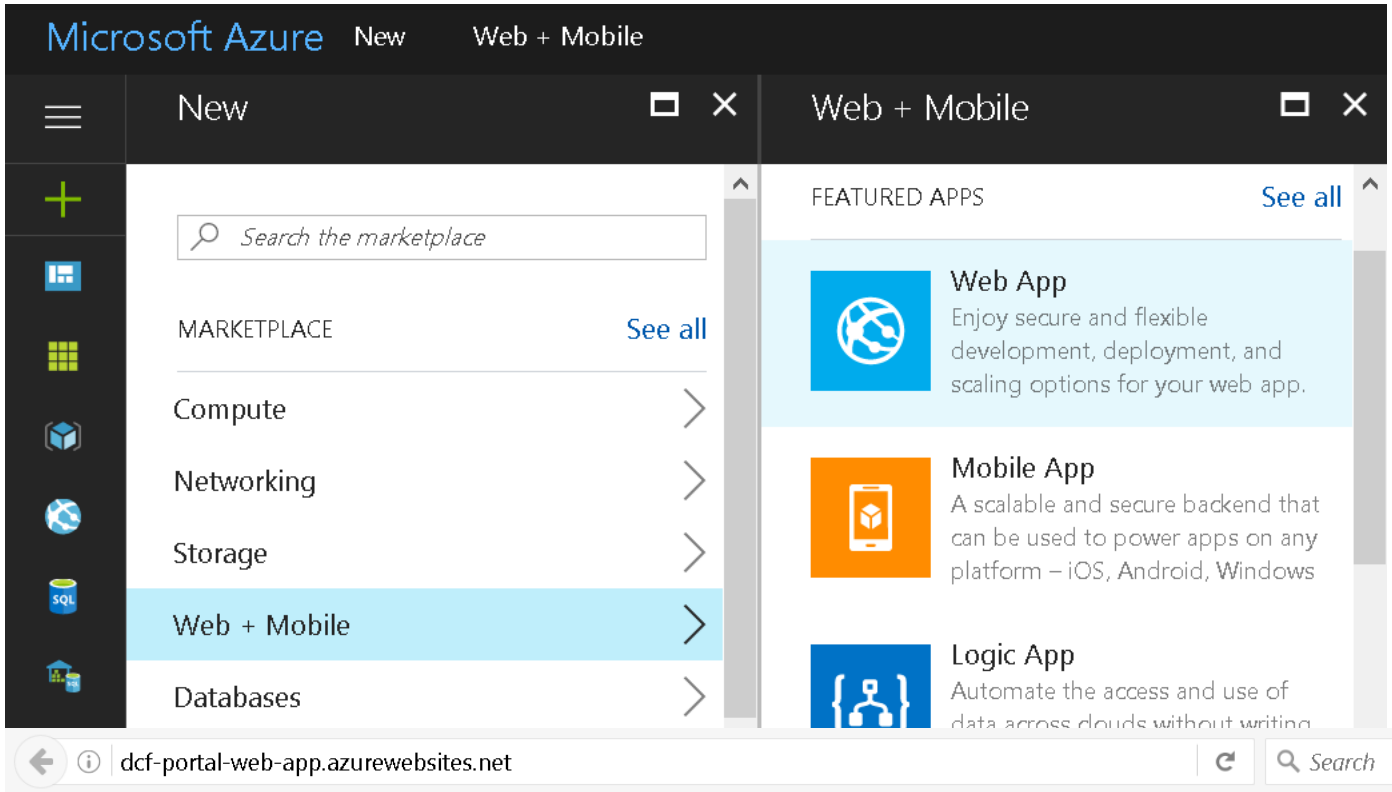
Mobile Apps

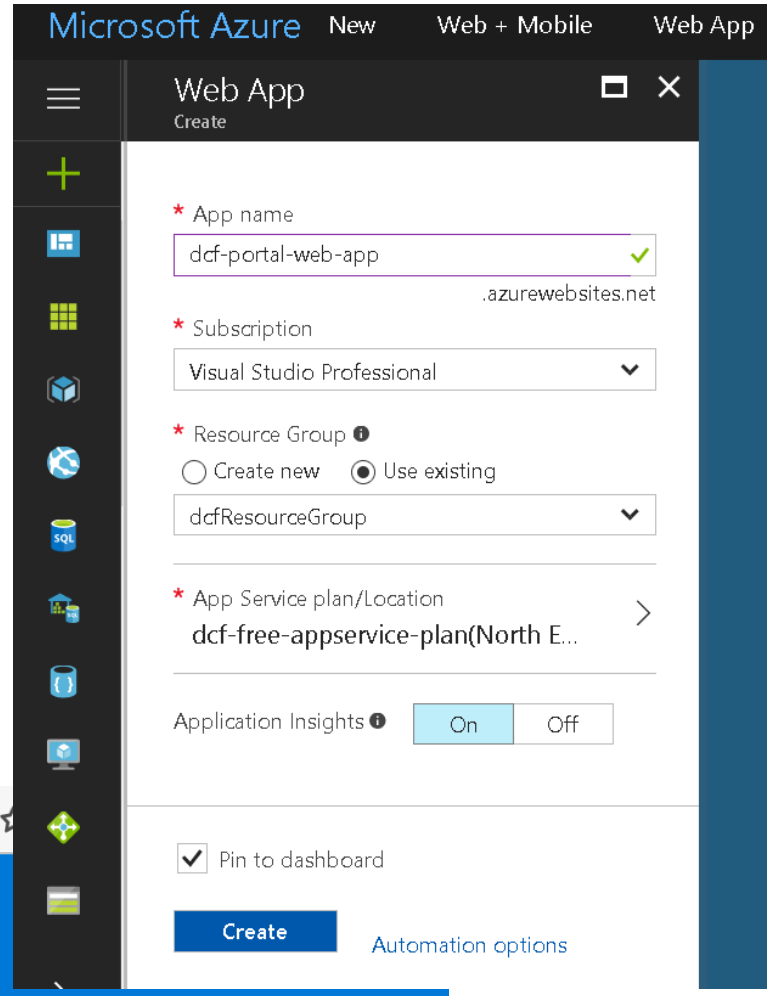
API Apps

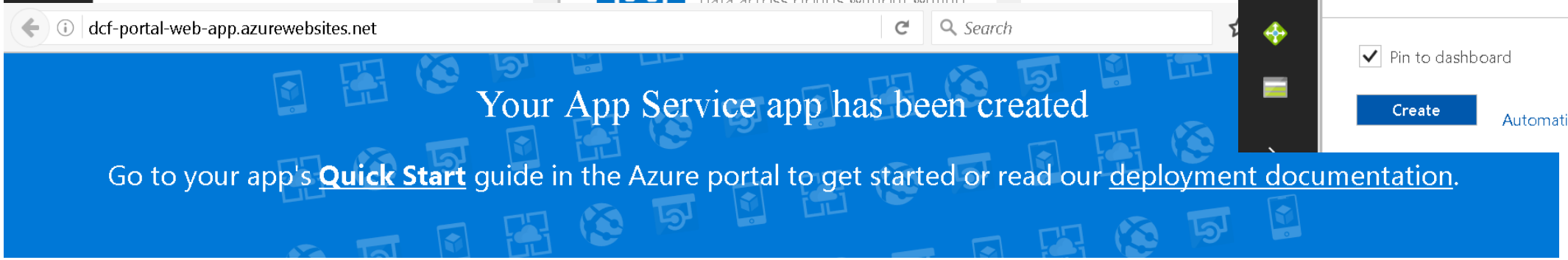
Logic Apps

AZURE WEB APPS VIA PORTAL

https://portal.azure.com







AZURE WEB APPS VIA AZURE CLI

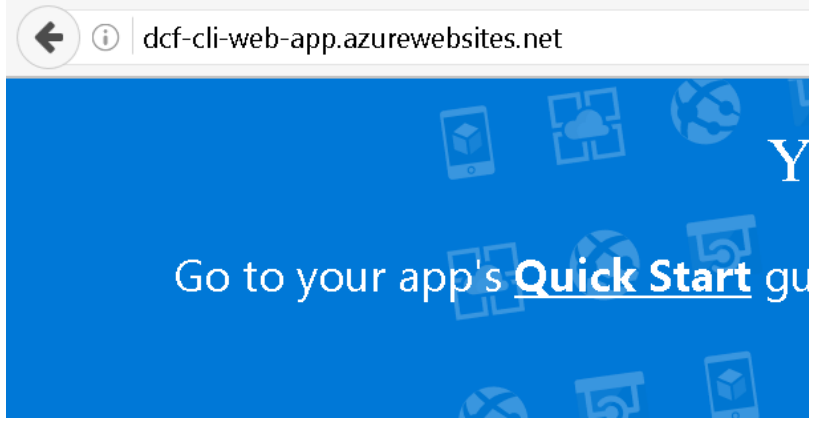
https://docs.microsoft.com/en-us/cli/azure/install-azure-cli

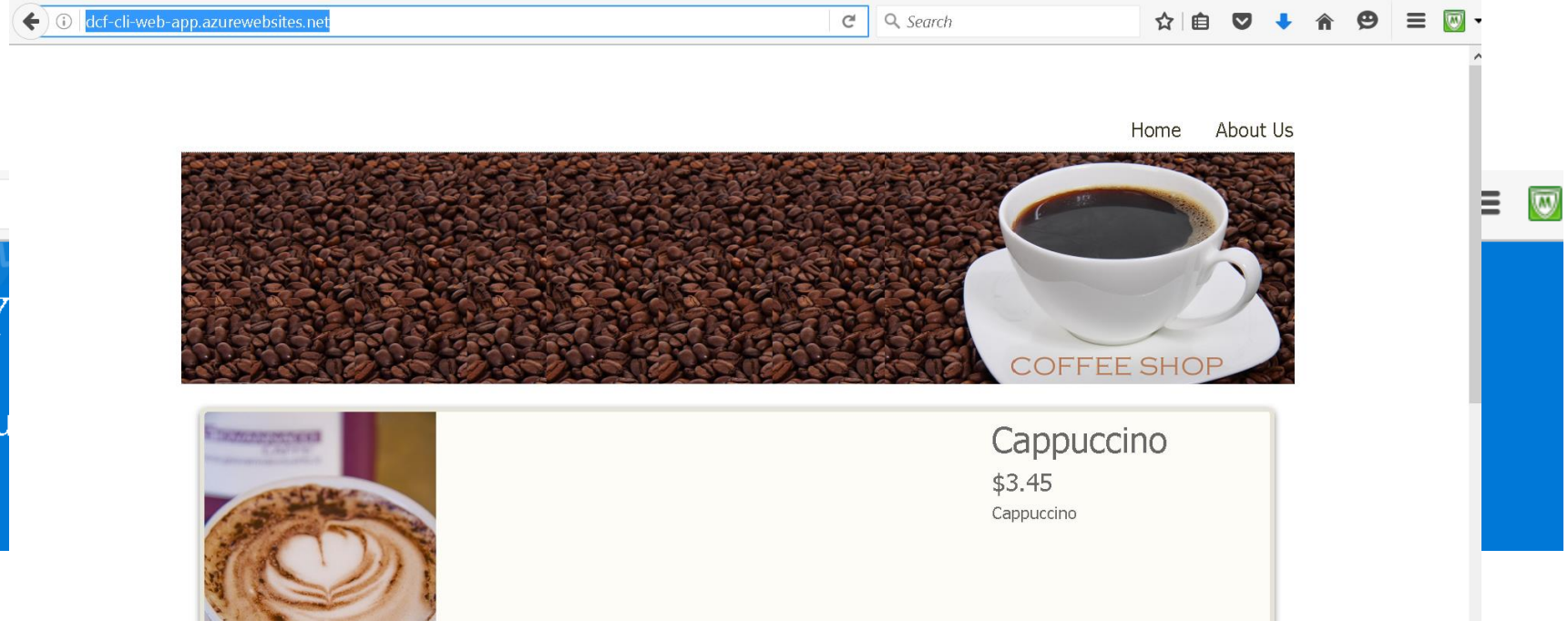
```
C:\Users\Dionne>az group create --location "North Europe" --name dcfResourceGroup
```

```
C:\Users\Dionne>az appservice plan create --name dcf-free-appservice-plan --resource-group dcfResourceGroup --sku FREE
```

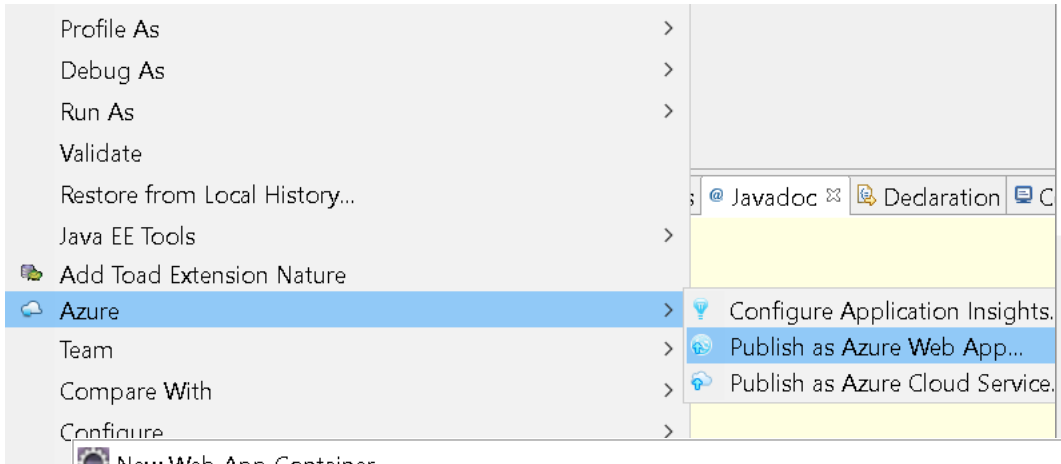
```
C:\Users\Dionne>az appservice web create --name dcf-cli-web-app --resource-group dcfResourceGroup --plan dcf-free-appservice-plan
```

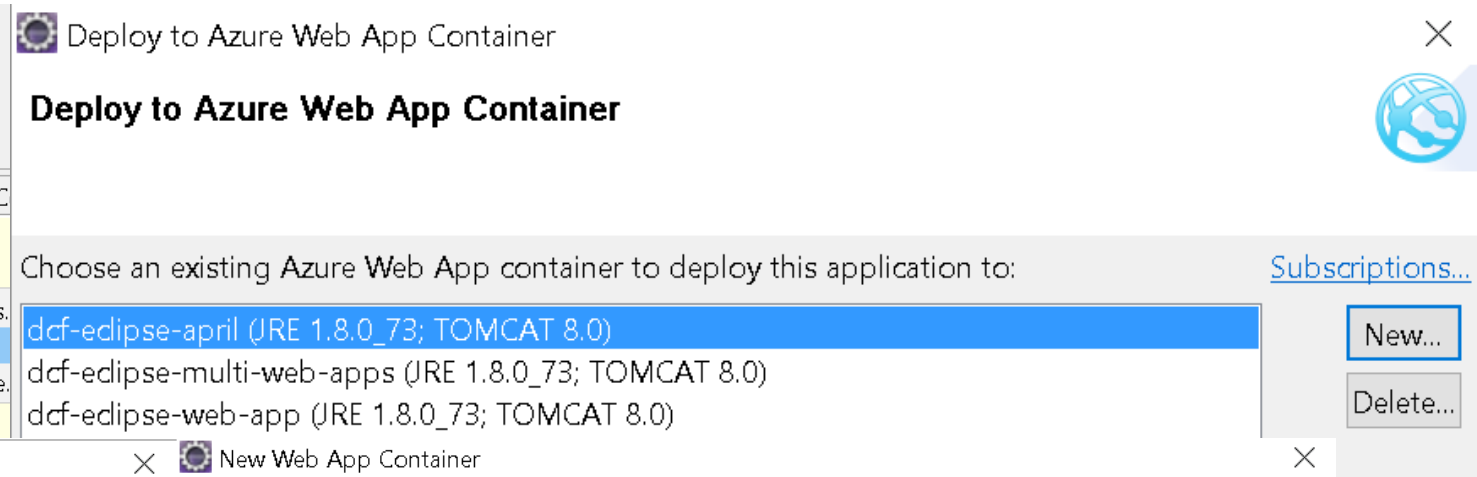
```
C:\Users\Dionne>az appservice web source-control config --name dcf-cli-web-app --resource-group dcfResourceGroup --repo-url "https://github.com/azure-appservice-samples/JavaCoffeeShopTemplate.git" --branch master --manual-integration
```

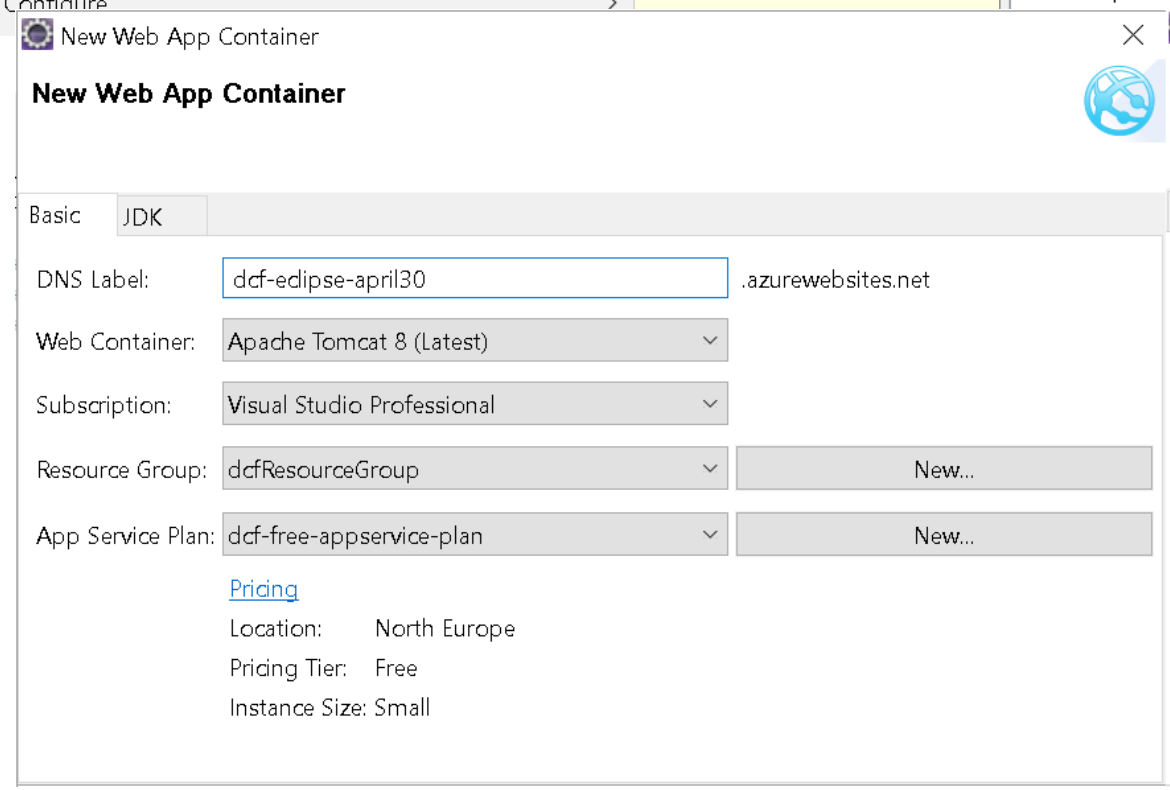


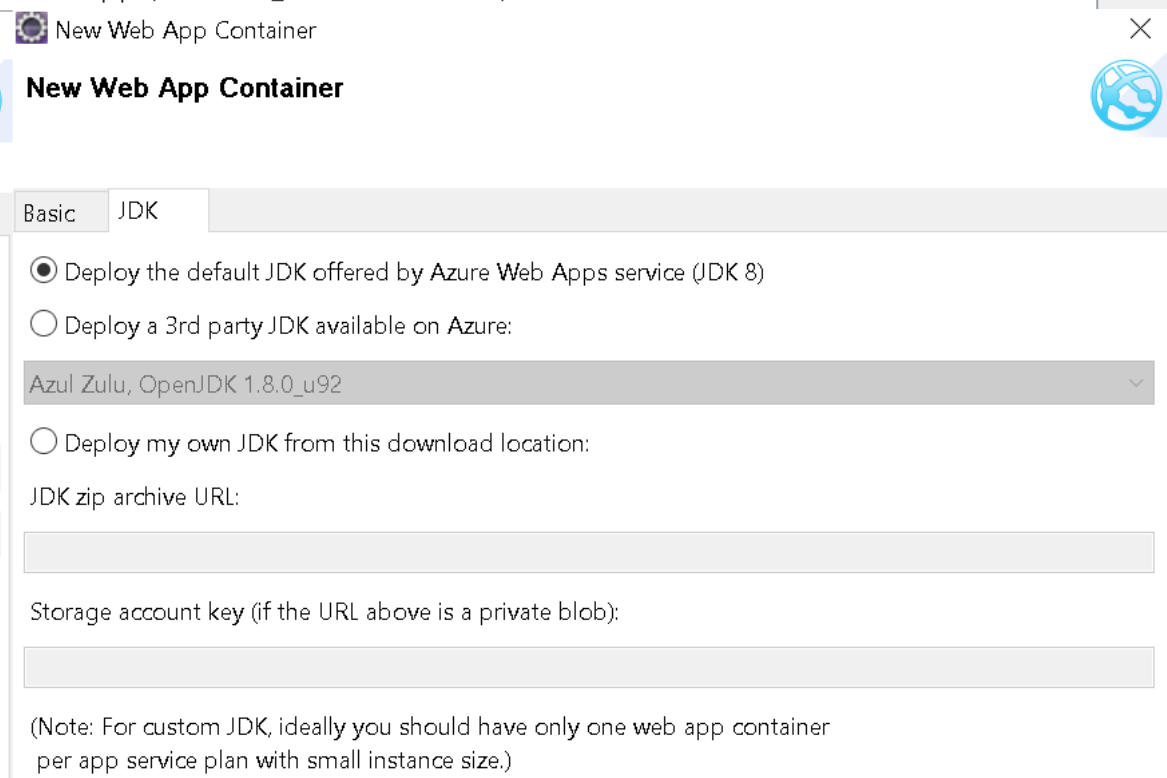


AZURE WEBAPPS VIA TOOLKIT FOR IDE









AZURE WEBAPPS DEPLOYMENT

Azure Toolkit for IDEs

FTP or Open Source Version Control Repositories or Private Repositories

Integrate with your favourite Continuous Integration/Continuous Deployment tools

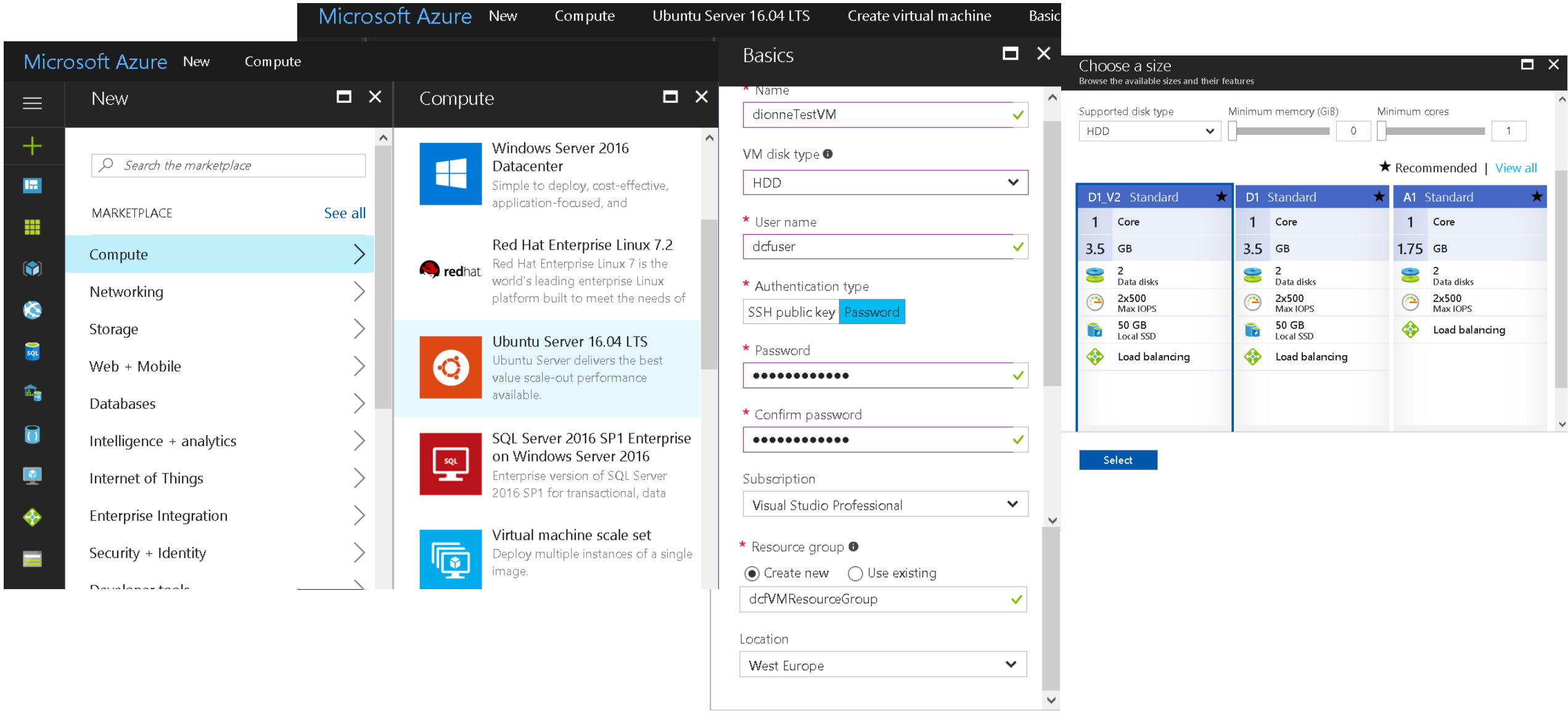
AZURE VIRTUAL MACHINES

- 1 in 3 Azure VMs are running Linux

  e.g. Centos, Suse, Core OS VMs, Red Hat, Ubuntu

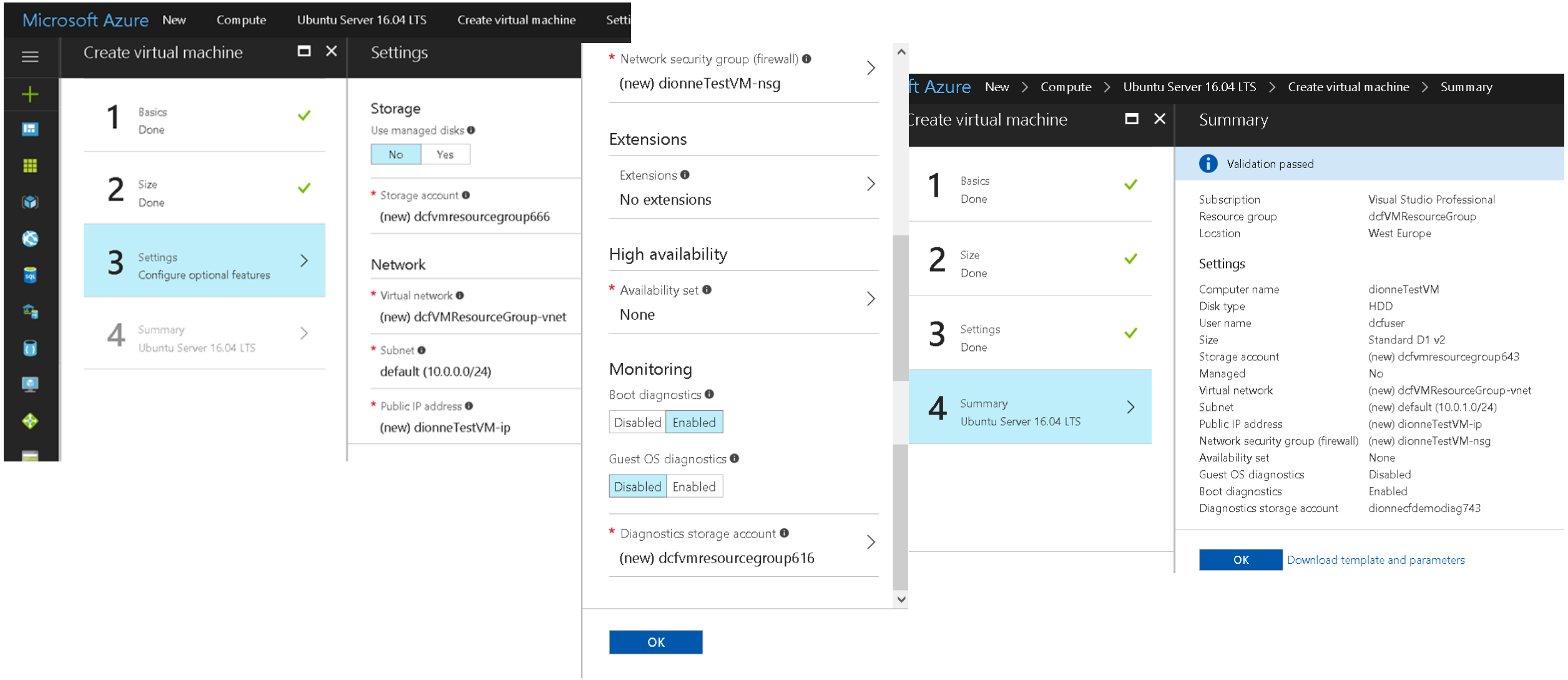
- Supports language like Node.JS, ASP.NET, Python, PHP, Java, Ruby
- Control over the Operating System, Software installation, Maintenance, Updates, Security, etc.
- Good for migrating existing applications to the Cloud Computing with “Lift and Shift” approach

AZURE LINUX VM – BASICS AND SIZE



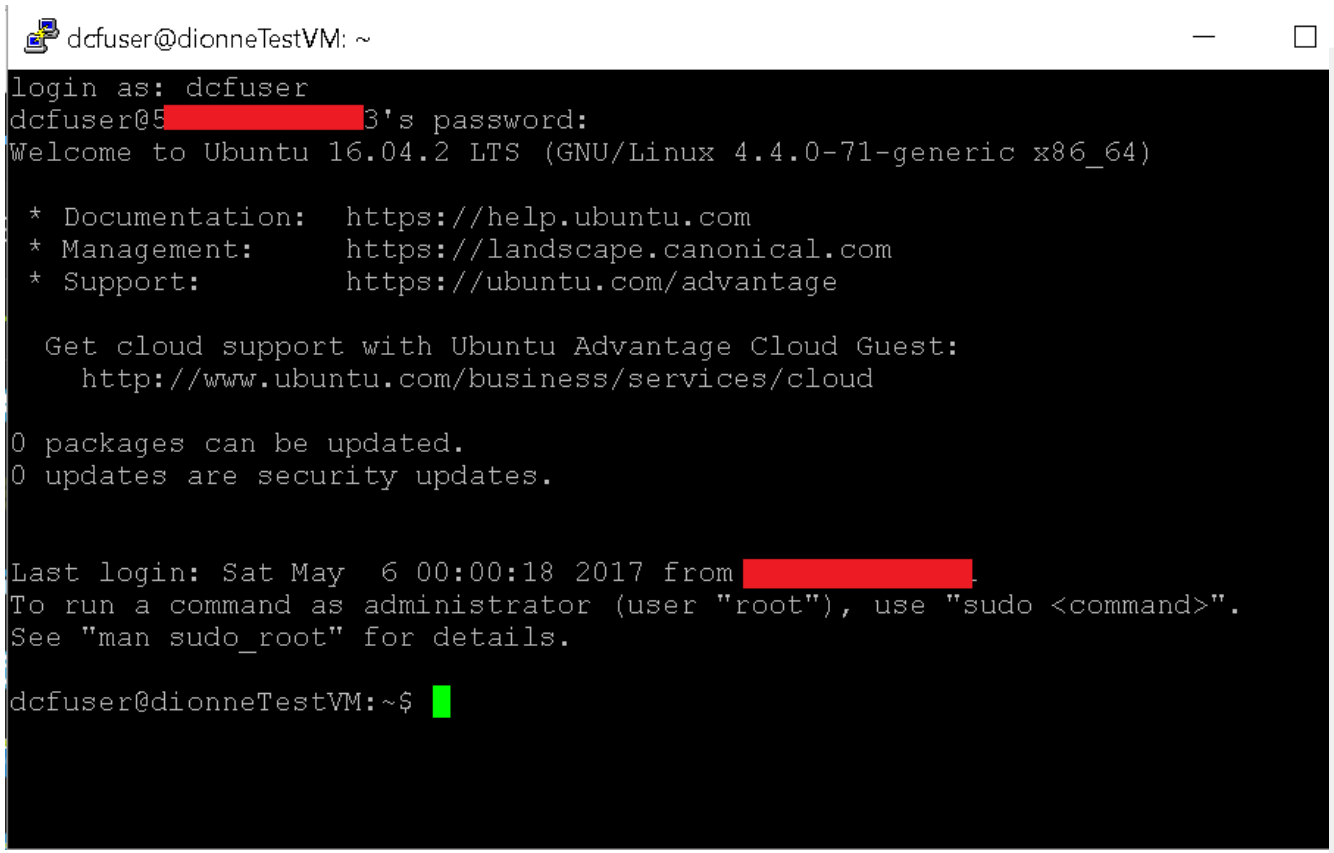
AZURE LINUX VM – VIA PORTAL

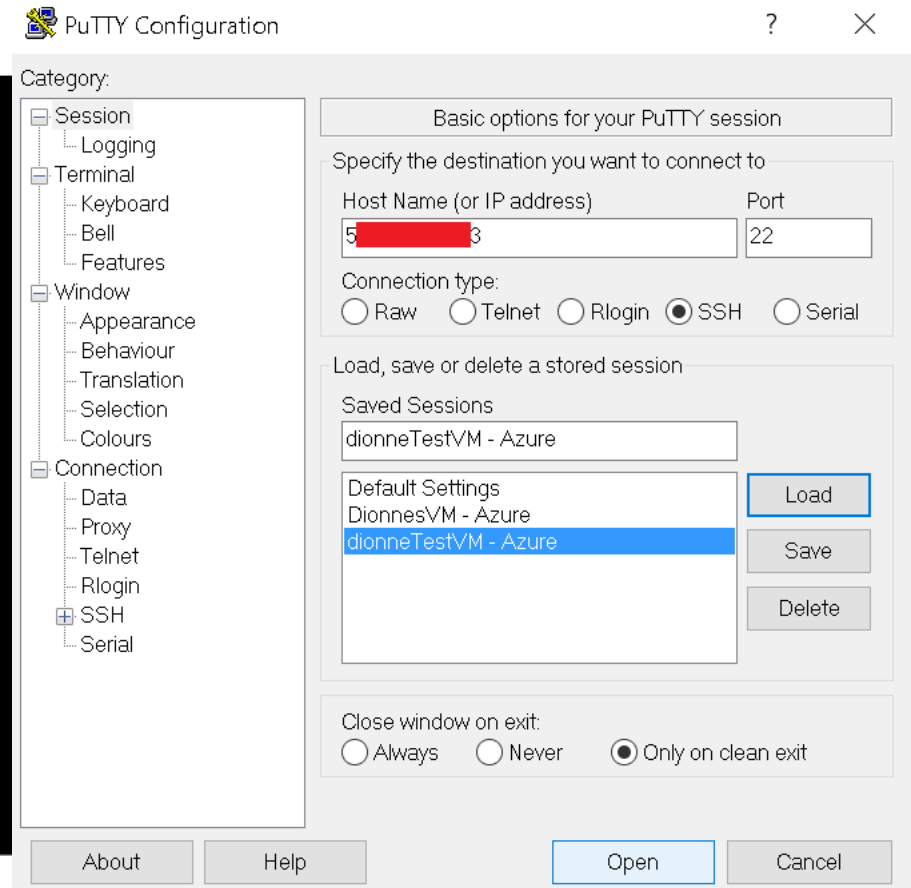
SETTINGS AND SUMMARY



AZURE LINUX VM - CONNECT VIA PUTTY (SSH)

```
C:\Users\Dionne>az vm list --output table
Name          ResourceGroup       Location
------------  ------------------  ----------
dionneTestVM  DCFVMRESOURCEGROUP  westeurope
DionnesVM     DIONNECFDEMO        westeurope
```



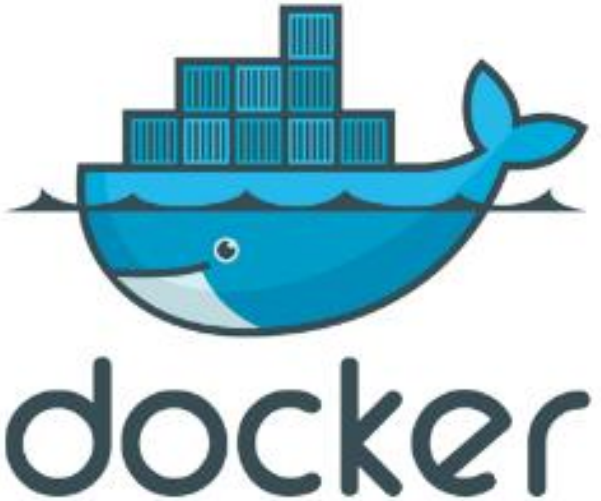


AZURE LINUX VM - SET UP

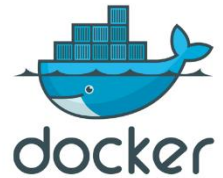
“Docker automates the repetitive tasks of setting up and configuring development environments so that developers can focus on what matters: building great software.

Developers using Docker don’t have to install and configure complex databases nor worry about switching between incompatible language toolchain versions. When an app is dockerized, that complexity is pushed into containers that are easily built, shared and run.”

docker.com



AZURE VM AND DOCKER
VIA DOCKER MACHINE AZURE DRIVER




```
C:\Users\Dionne>docker-machine create -d azure --azure-ssh-user ops --azure-subscription-id 7[redacted]0c --azure-resource-group dcfVMDockerMachine
Group --azure-location westeurope --azure-open-port 80 dcfmachine
```

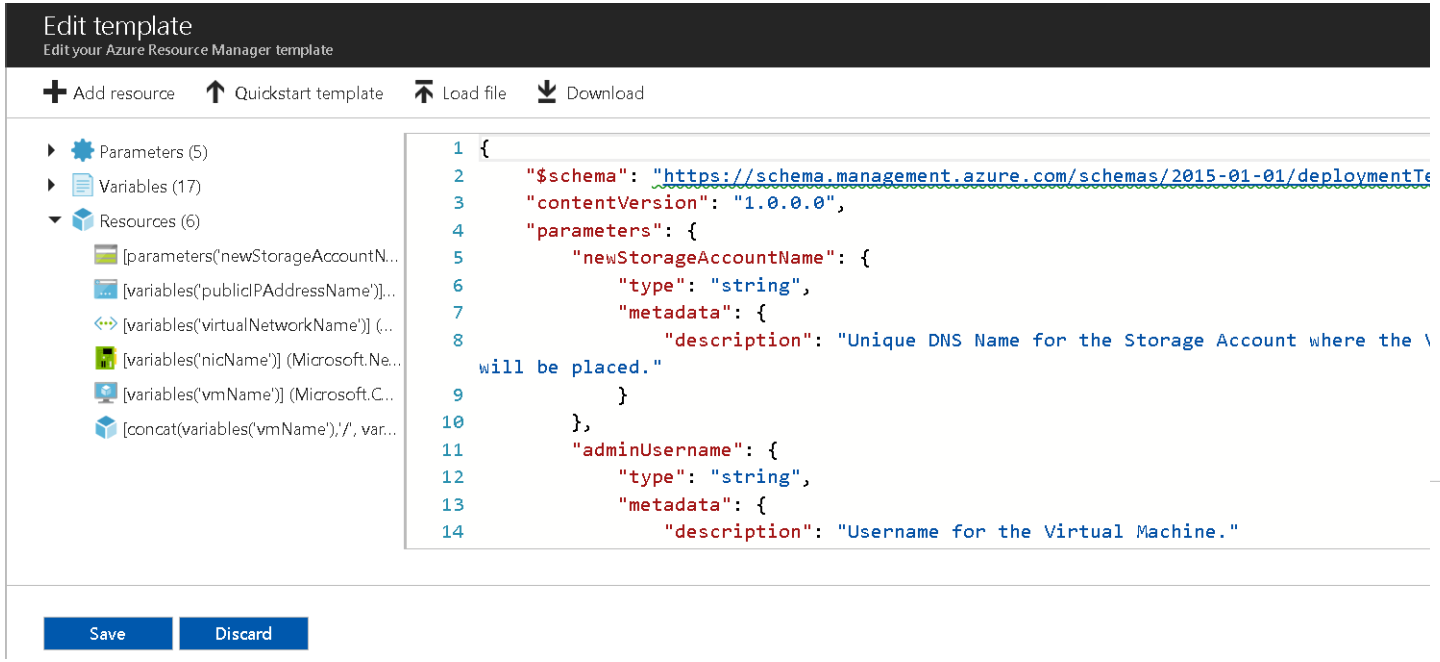
```
Running pre-create checks...
(dcfmachine) Completed machine pre-create checks.
Creating machine...
(dcfmachine) Querying existing resource group.  name="dcfVMDockerMachineGroup"
(dcfmachine) Creating resource group.  name="dcfVMDockerMachineGroup" location="westeurope"
(dcfmachine) Configuring availability set.  name="docker-machine"
(dcfmachine) Configuring network security group.  location="westeurope" name="dcfmachine-firewall"
(dcfmachine) Querying if virtual network already exists.  name="docker-machine-vnet" rg="dcfVMDockerMachineGroup" location="westeurope"
(dcfmachine) Creating virtual network.  location="westeurope" name="docker-machine-vnet" rg="dcfVMDockerMachineGroup"
(dcfmachine) Configuring subnet.  name="docker-machine" vnet="docker-machine-vnet" cidr="192.168.0.0/16"
(dcfmachine) Creating public IP address.  static=false name="dcfmachine-ip"
(dcfmachine) Creating network interface.  name="dcfmachine-nic"
(dcfmachine) Creating storage account.  name="vhds4lofa5lso94q2weg40qu" location="westeurope" sku=Standard_LRS
(dcfmachine) Creating virtual machine.  location="westeurope" size="Standard_A2" username="ops" osImage="canonical:UbuntuServer:16.04.0-LTS:latest" name="dcfmachine"
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with ubuntu(systemd)...
Installing Docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: docker-machine env dcfmachine
```

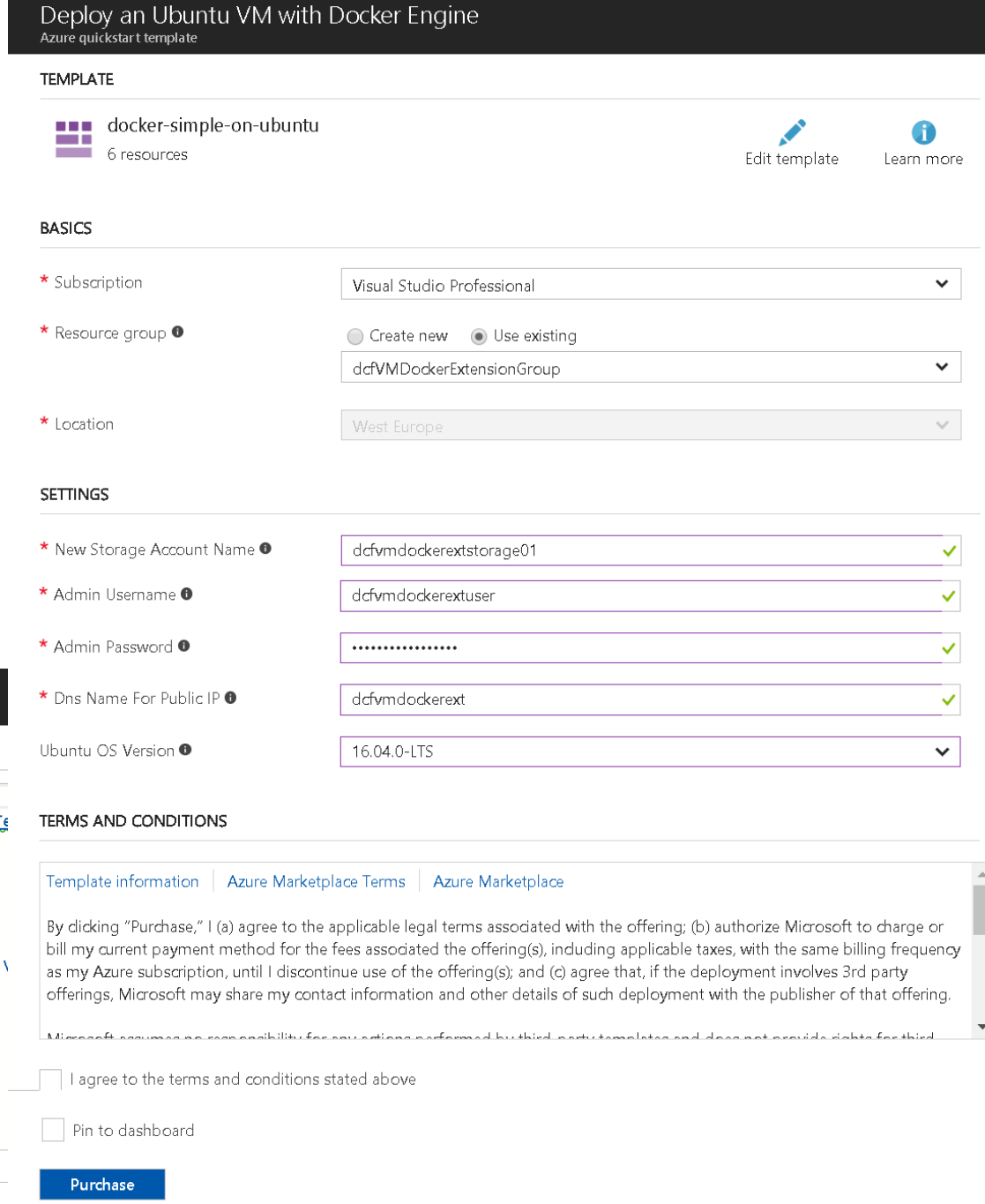
AZURE VM AND DOCKER EXTENSION VIA ARM Templates

https://portal.azure.com/#create/Microsoft.Template

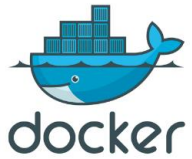
Azure QuickStart Template

https://github.com/Azure/azure-quickstart-templates/tree/master/docker-simple-on-Ubuntu/azuredeploy.json





DOCKER EXAMPLE



- **Pre-built Image - Oracle XE 11g** https://hub.docker.com/r/wnameless/oracle-xe-11g

  docker pull wnameless/oracle-xe-11g

  docker run –d –p 49160:22 –p 49161:1521 –p 49162:8080 wnameless/oracle-xe-11g

- **Custom Image – Based on JBOSS Wildfly** https://hub.docker.com/r/jboss/wildfly

1) Create a Dockerfile

```
FROM jboss/wildfly
ADD hello.war  /opt/wildfly/standalone/deployments/
```

2) Build the Dockerfile to create an image: ***docker build --tag=wildfly-app .***

3) Run container based on the built images: ***docker run -it -p 8080:8080 wildfly-app***

4) Check container is running: ***docker ps***

DOCKER EXAMPLE VIA COMPOSE

~$vi docker-compose.yml

```yaml
version: '3'
services:
  springwebapp:
    build:
      context: .
      dockerfile: springwebapp.Dockerfile
    image: dionnecf/springwebapp
    ports:
      - "8080:8080"
    networks:
      - net-todo
    volumes:
    - .:/vol/development
    depends_on:
      - dcfmysqlserver
    container_name: springwebapp

  dcfmysqlserver:
    build:
      context: .
      dockerfile: springmysqldb.Dockerfile
    image: dionnecf/dcfmysqlserver
    ports:
      - "3306:3306"
    networks:
      - net-todo
    environment:
      MYSQL_DATABASE: tododb
      MYSQL_USER: dcfuser
      MYSQL_PASSWORD: dcfpassword
      MYSQL_ROOT_PASSWORD: rootpassword
    container_name: dcfmysqlserver

networks:
  net-todo:
    driver: bridge
```

Create and start containers

~$docker-compose up –d

which containers are running

~$docker-compose ps

```
$ docker-compose ps
     Name                   Command             State                  Ports
-------------------------------------------------------------------------------------
dcfmysqlserver   /entrypoint.sh mysqld          Up      0.0.0.0:3306->3306/tcp, 33060/tcp
springwebapp     mvn clean package exec:java    Up      0.0.0.0:8080->8080/tcp
```

Check webapp works

```
$ curl -H "Content-Type: application/json" -X GET "http://192.168.99.100:8080/"
<h1>Hi, Welcome to my ToDoList app running in Docker with Mysql... :)</h1>
```

Check webapp retrieves data from database

```
$ curl -H "Content-Type: application/json" -X GET "http://192.168.99.100:8080/todos"
[{"id":1,"itemName":"Train Travel","itemDesc":"Remember to purchase your train ticket","createdDate":1491436800
te":1491436800000,"completed":false},{"id":3,"itemName":"Snacks","itemDesc":"Remember to purchase some snacks f
k up your dry cleaning before you go","createdDate":1491436800000,"completed":false}]
```

Tail the logs for a container

~$docker-compose logs springwebapp -f

DOCKFILE FOR CUSTOM IMAGES

~$ vi springwebapp.Dockerfile

```
FROM java:8

MAINTAINER Dee Farrell <dee@dcfhosting.co.uk>

# update packages and install maven
RUN  \
  export DEBIAN_FRONTEND=noninteractive && \
  sed -i 's/# \(.*multiverse$\)/\1/g' /etc/apt/sources.list && \
  apt-get update && \
  apt-get -y upgrade && \
  apt-get install -y vim wget curl maven

# attach volumes
VOLUME /vol/development

# create working directory
RUN mkdir -p /vol/development
WORKDIR /vol/development

# maven exec
CMD ["mvn", "clean", "package", "exec:java"]
```

~$ vi springdbserver.Dockerfile

```
FROM mysql/mysql-server

MAINTAINER Dee Farrell <dee@dcfhosting.co.uk>

ADD data/ /docker-entrypoint-initdb.d/
```

Life Before Docker

Stressed Out!

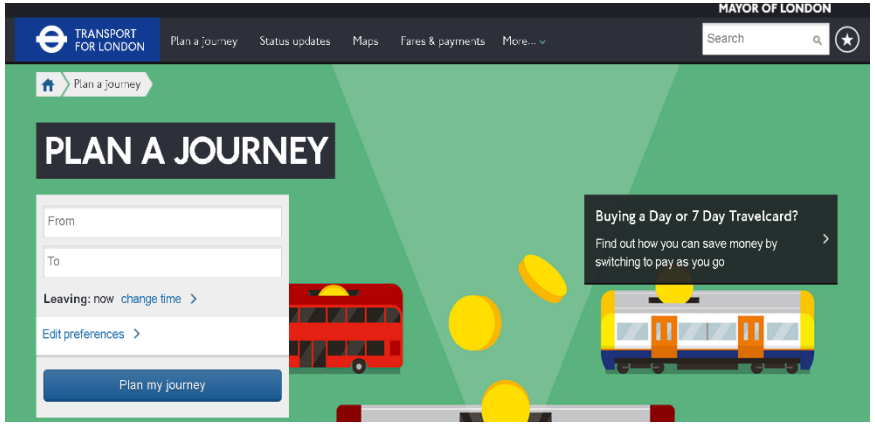
Life Since Docker

Stress Free and Relaxed

TFL AND THE CLOUD

AZURE

AWS



AWS

AZURE

TFL AND THE CLOUD

- TfL Wide Application Hosting Strategy currently being planned
- Cloud First approach, followed by Hybrid Cloud/On-premises approach
- Initial review shows over 50% of existing applications ready for cloud hosting (i.e. lift and shift).
- Deeper dive discovery is planned to confirm.

JOURNEY TO THE CLOUD

A Developer's Perspective

- Cloud Computing provides time savings, simplicity, flexibility for Developers
- Many Cloud Computing Providers out there to learn about
- Get started with Azure App Service and Azure Virtual Machines
- Save your sanity, just use Azure VMs with Docker and Compose
- Give it a go…. Cloud computing is not as complicated as it first seems

https://azure.microsoft.com/en-us/try/app-service/

https://azure.microsoft.com/en-gb/campaigns/developer-guide/