Deploying Confluent Enterprise on Microsoft Azure

An Introductory Guide

April 2017

ABSTRACT

This white paper outlines the integration of Confluent Enterprise with Microsoft's Azure Cloud Platform. The Confluent Enterprise offer within Azure Marketplace is presented along with recommended best practices for integrating with other Azure services.

Table of Contents

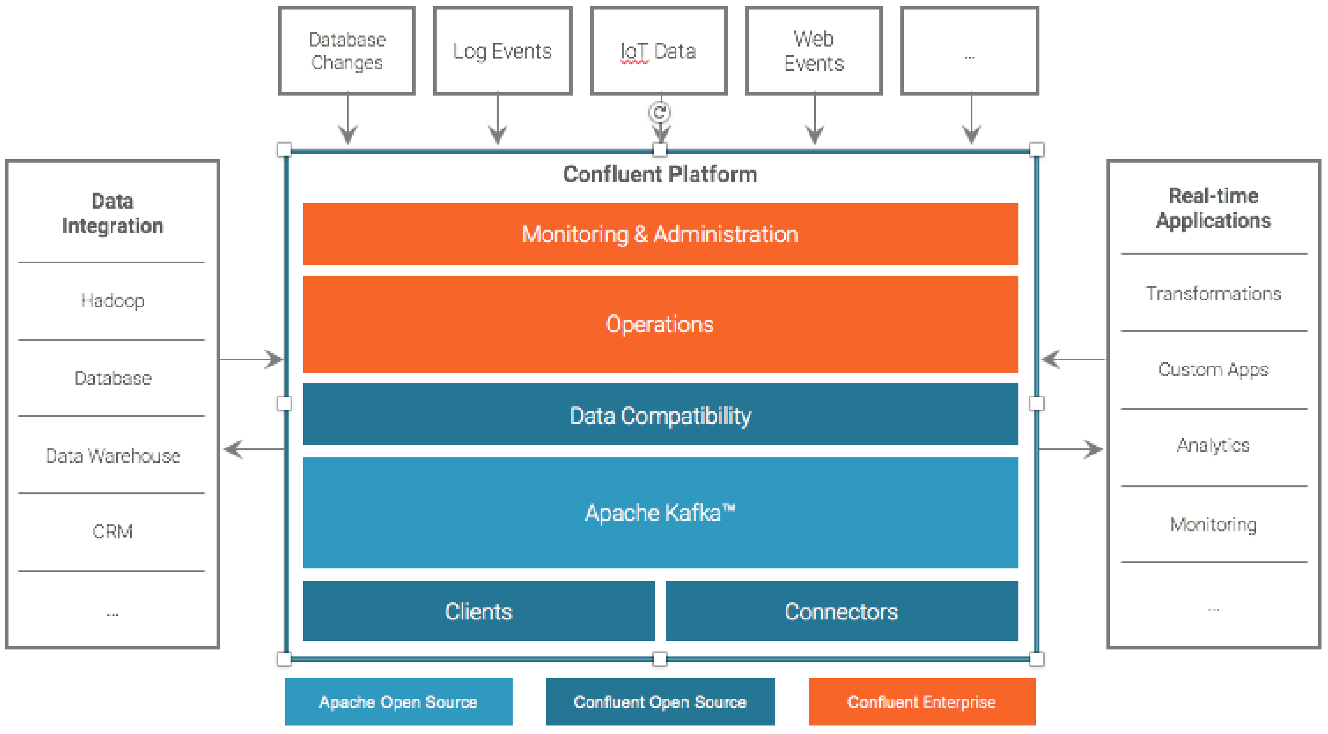
Confluent and Azure

Apache Kafka™, created by the founders of Confluent while working at LinkedIn, is fast becoming the de facto technology for moving data across systems in real time. Such frictionless data movement supports interactive websites, mobile applications, and all manner of Internet-of-Things use cases. Confluent builds on Kafka with additional tooling to speed developer productivity, improve data management, and broaden the scope of applications that can benefit from Kafka.

Confluent Enterprise is available as a bring-your-own-license offering in Azure Marketplace. The integration simplifies deployment of your streaming infrastructure and applications, making it easy to configure, deploy, monitor and manage Confluent clusters. Kafka clusters on Azure can support 10s to 1000s of traditional producer/consumer workloads as well as the latest Kafka Connect pipelines.

Confluent Enterprise

The Confluent architecture is summarized in the following diagram:



At the center of Confluent Enterprise is Apache Kafka. Apache Kafka is a modern distributed system for moving data in real time. It allows a single cluster of computers to perform all the data transport between an organization's data systems regardless of their number or size. Confluent Enterprise includes all components from the Apache Kafka Core along with some Apache-licensed open source additions (client tools, pre-built connectors, and cluster-side services such as Schema Registry and REST Proxy). The offering is completed with licensed components to support cluster monitoring and management, multi-data center replication, and advanced data balancing capabilities.

Azure Services

With Azure Marketplace, you can provision big data services with a single click. Services can leverage the same virtual infrastructure to achieve tight integration, or remain loosely coupled though limited network or storage connectivity. The Azure Marketplace offering for Confluent Enterprise delivers

- Automated cluster provisioning, management, and elastic scaling
- Complete support for all Confluent Enterprise components, e.g. Kafka, Kafka Connect Workers, Kafka Streams Apps, Schema Registry, REST Proxy, and Control Center
- Point-and-click data pipeline deployment between other Azure services via the Kafka Connect.

Confluent Enterprise Quick Start on Azure

Initial support for Confluent Enterprise 3.0 was delivered in October 2016, with an upgrade to later versions of Confluent 3.x April 2017. The services are deployed across a user-defined set of virtual machine instances that can be sized to meet your specific needs. These installation instructions assume that you have an active Azure Marketplace subscription and a basic understanding of the Confluent architecture (the relationship between brokers, workers, and support services {Schema Registry and REST Proxy}). More information on the Confluent architecture is available here.

Topology overview

For simplicity, the Confluent Enterprise topology supports only three classes of service nodes for the deployment. Users will specify 3 or more broker nodes and 1 or more worker nodes. Apache Kafka requires a zookeeper quorum for metadata management; users can specify independent nodes for the zookeeper service, or allow the service to be hosted on the broker nodes. Auxiliary services (Confluent Schema Registry and Confluent REST Proxy) are deployed on the worker nodes.

All standard virtual machine instance types are supported. The broker nodes are automatically provisioned with 4 TB of storage each. All instances will be deployed within a single virtual subnet. Users can select a pre-existing network within their Azure subscription or create a new one during the deployment process. By default, the deployed nodes will be assigned public hostnames / IP addresses and the network ports to access Confluent Enterprise services from outside Azure will be opened for external access. The specifics of that access mechanism is discussed later in this document.

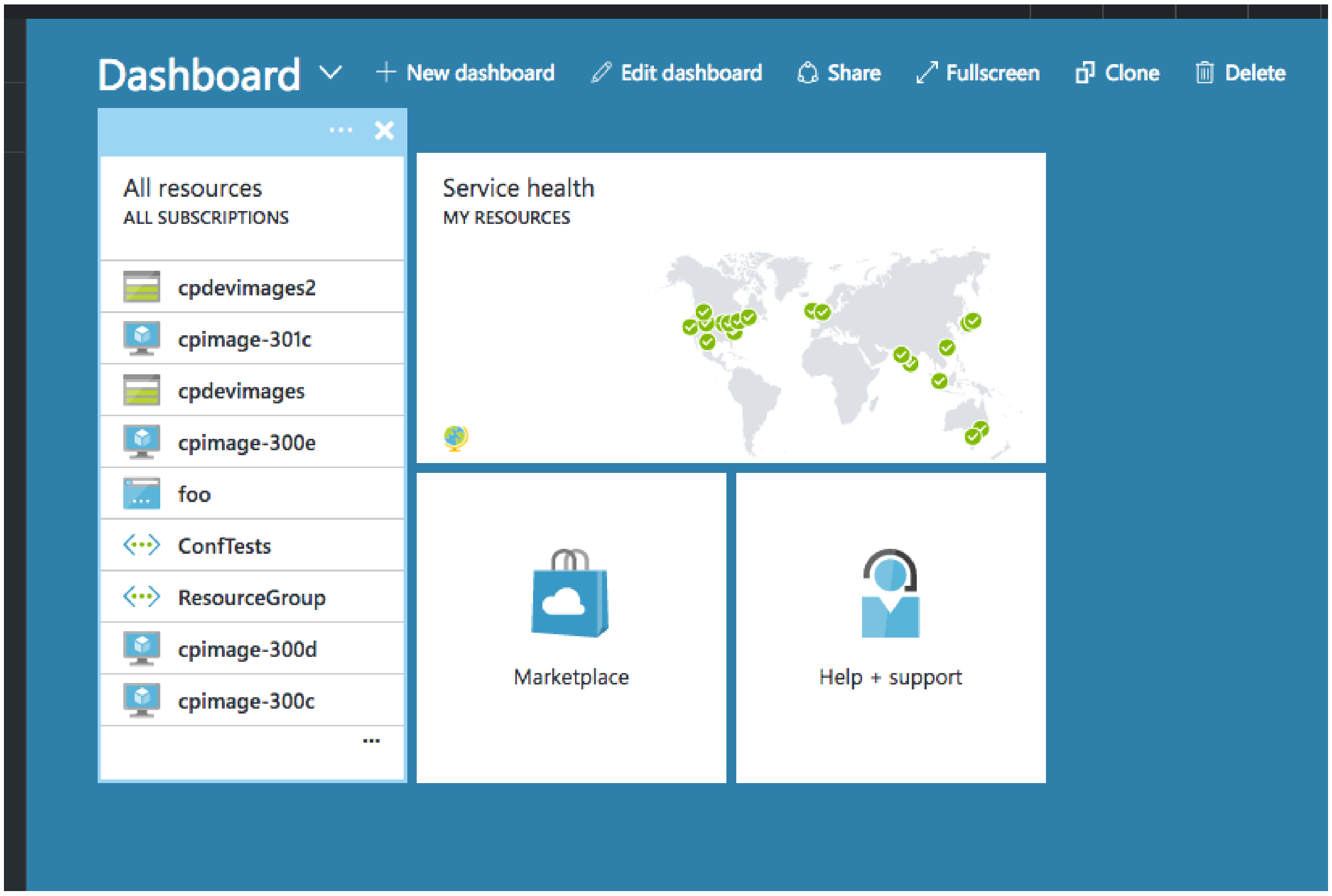
The following sections discuss the deployment process in a step-by-step manner.

Download and install the Azure command line tool

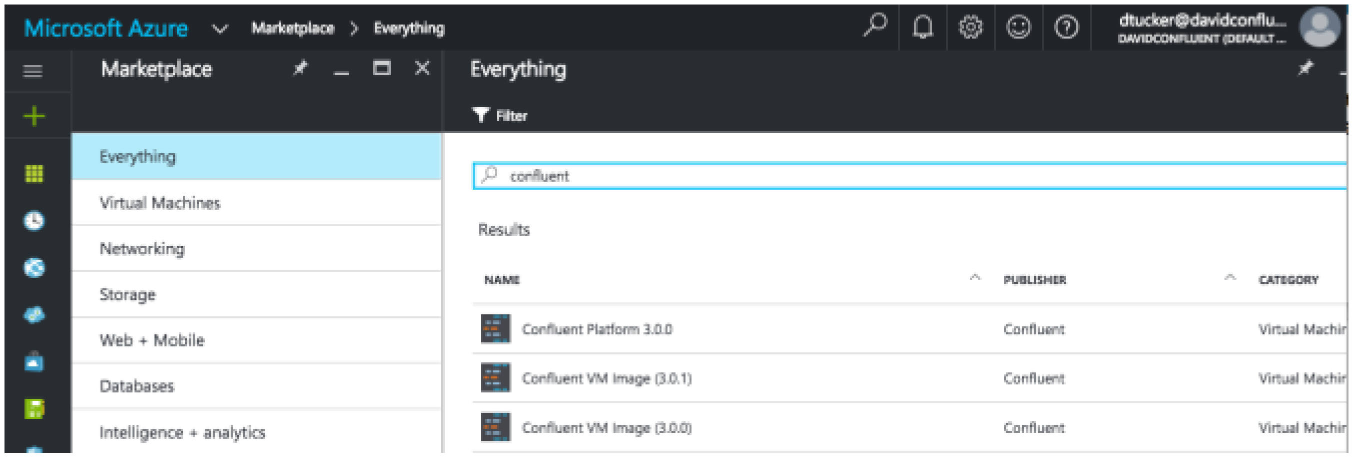
There are occasions when command-line control over your Azure services is useful. For that reason, you may wish to install the Azure command-line tool by following the instructions found at http://azure.microsoft.com/en-us/documentation/articles/xplat-cli/#configure . Be sure to authenticate your client with the Azure portal before proceeding with the rest of the deployment.

Connect to the Azure portal

The Azure portal is available at http://portal.azure.com . After logging in, your browser will show the top-level dashboard :



Click on the Marketplace link to navigate to the services page. Select the “Compute” category and then search for the Confluent offerings. Select the desired Confluent Enterprise version and continue with the deployment.



Results

NAME	PUBLISHER	CATEGORY
Confluent Platform 3.0.0	Confluent	Virtual Machir
Confluent VM Image (3.0.1)	Confluent	Virtual Machir
Confluent VM Image (3.0.0)	Confluent	Virtual Machir

NOTE: There can be some confusion in the Services page between the standalone Confluent VM Images and the complete Azure Topology that will deploy multiple images into a single, coherent cluster. Be sure to select the Confluent Enterprise solution topology link, not any of the VM Image links. If the input blades do not match the example below, it's likely that you selected the VM Image only.

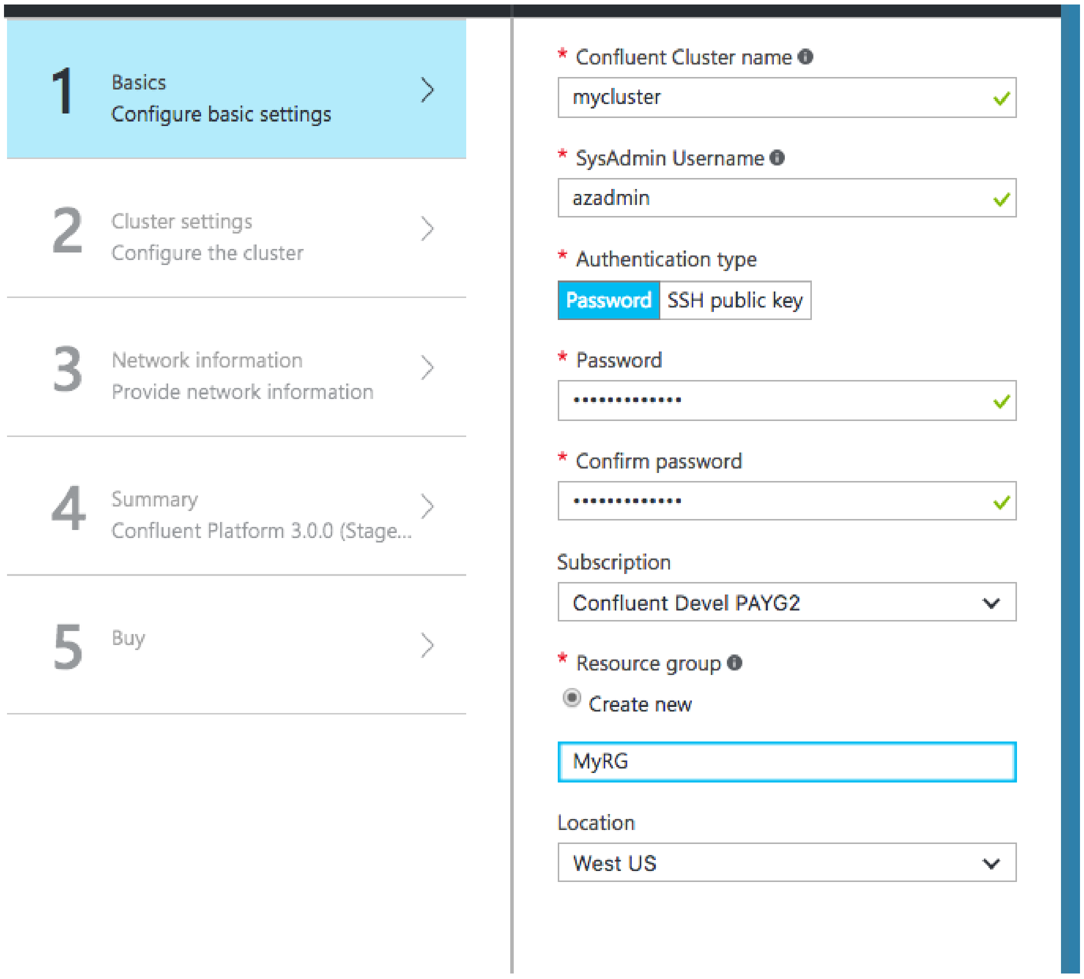
The solution template allows you to deploy any of the Confluent versions available in Azure.

Deploy Cluster

The deployment process consists of a series of blades where you will specify the necessary parameters to configure your cluster. Each field within the blade will have a tool-tip to assist you, as well as a reasonable default value (when possible)

I. Basics

The first blade consists of the cluster name and system administrator access credentials. You'll also specify the Resource Group for the infrastructure and the Azure region for the deployment.



* Confluent Cluster name: mycluster

* SysAdmin Username: azadmin

* Authentication type: [✓ Password] SSH public key

* Password: ••••••••••••••

* Confirm password: ••••••••••••••

Subscription: Confluent Devel PAYG2

* Resource group: [✓ Create new] MyRG

Location: West US

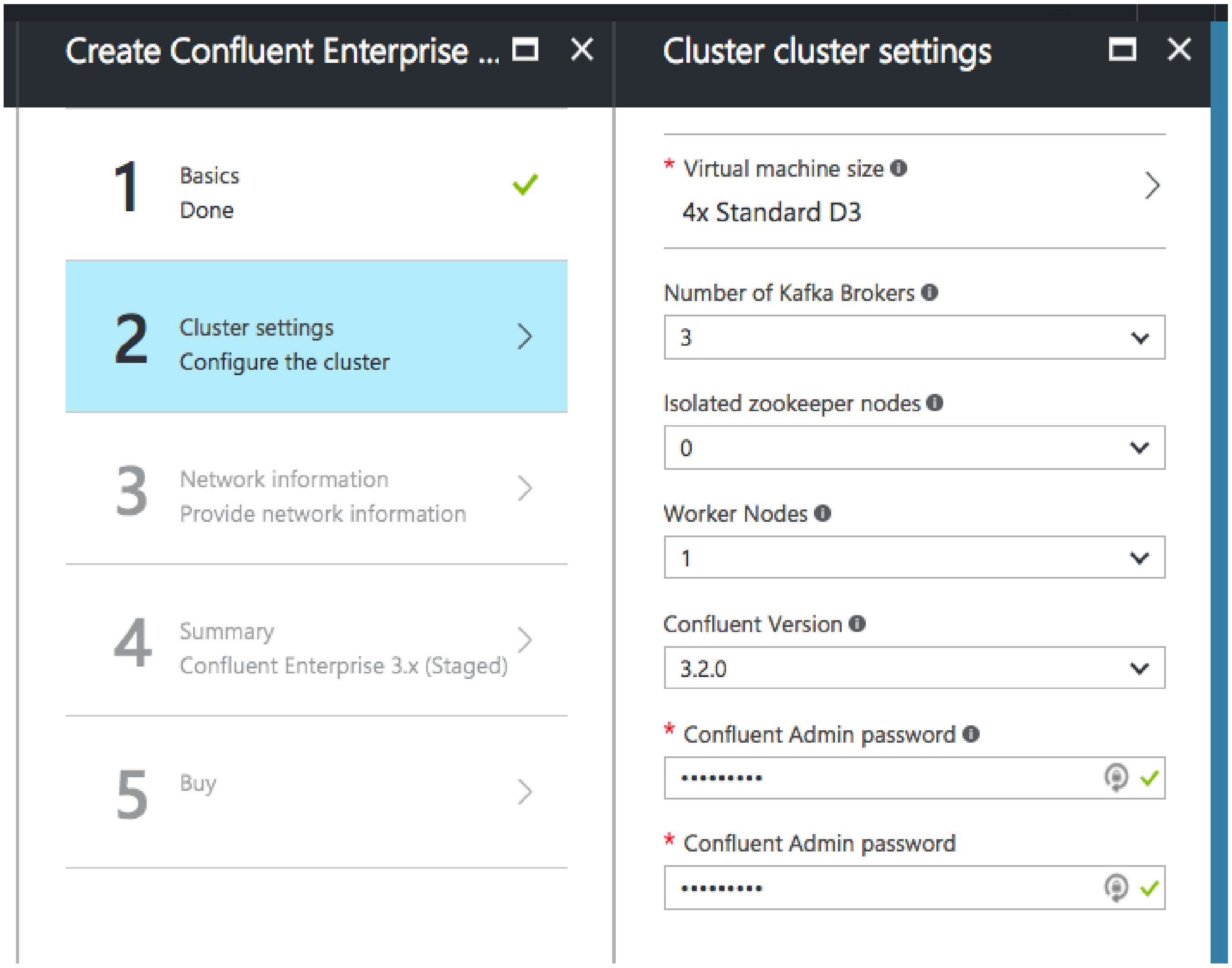
The Cluster name will be used as a seed for the deployed resources (virtual machines, hostnames, storage accounts, etc.). Within any single resource group, you should not deploy two Confluent Enterprise clusters of the same name.

The SysAdmin specifications define a Linux user and authentication type for the server administrator. That user will have pseudo privileges and will be able to manage the server from an operational perspective. While you can choose simple password authentication, Microsoft and Confluent recommend using SSH Public Key authentication and providing the public half of your key-pair for the deployment.

As with other solution deployments, you must specify the data center location and the resource group into which the infrastructure will be provisioned. At this time, Azure does not support deploying 3rd party solutions such as Confluent Platform into an existing resource group.

II. Cluster settings

The next blade specifies the virtual machine details for the deployment: which machine size you prefer and how many hosts for the cluster overall.



Create Confluent Enterprise ...

1 Basics
Done ✓

2 Cluster settings
Configure the cluster

3 Network information
Provide network information

4 Summary
Confluent Enterprise 3.x (Staged)

5 Buy

Cluster cluster settings

* Virtual machine size
4x Standard D3

Number of Kafka Brokers: 3

Isolated zookeeper nodes: 0

Worker Nodes: 1

Confluent Version: 3.2.0

* Confluent Admin password: •••••••••

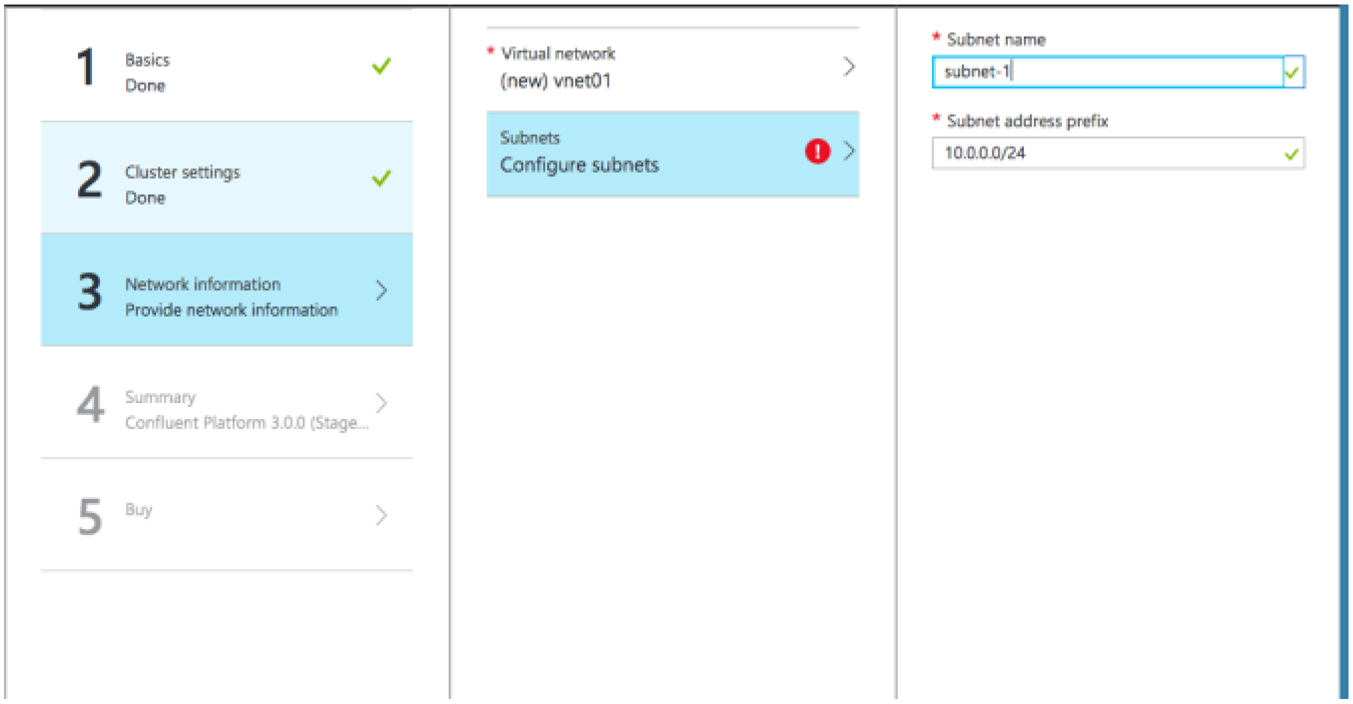
* Confluent Admin password: •••••••••

At a minimum, you cluster will have 3 broker nodes and 1 worker node. The zookeeper quorum can be deployed separately on 1 or 3 nodes, or co-resident with the brokers (when “0” zookeeper nodes are selected). This last case is perfectly reasonable for development environments, but may not be appropriate for high-volume clusters. For maximum performance and resiliency, Confluent recommends deploying 3 zookeeper nodes independent from the Kafka brokers.

The ***password*** setting in this blade is for the Confluent Enterprise administrator (Linux user: kadmin). All Confluent Enterprise services are executed as that user within the cluster. Remember that if the ***Authentication Type*** setting from the previous screen is set to **SSH Public Key**, password authentication to the kadmin account will be disabled.

III. Network information

The Confluent Enterprise topology allows you to create a completely new virtual network for the cluster, or select a pre-existing network.


1 Basics Done
2 Cluster settings Done
3 Network information Provide network information
4 Summary Confluent Platform 3.0.0 (Stage...
5 Buy

* Virtual network
(new) vnet01

Subnets
Configure subnets

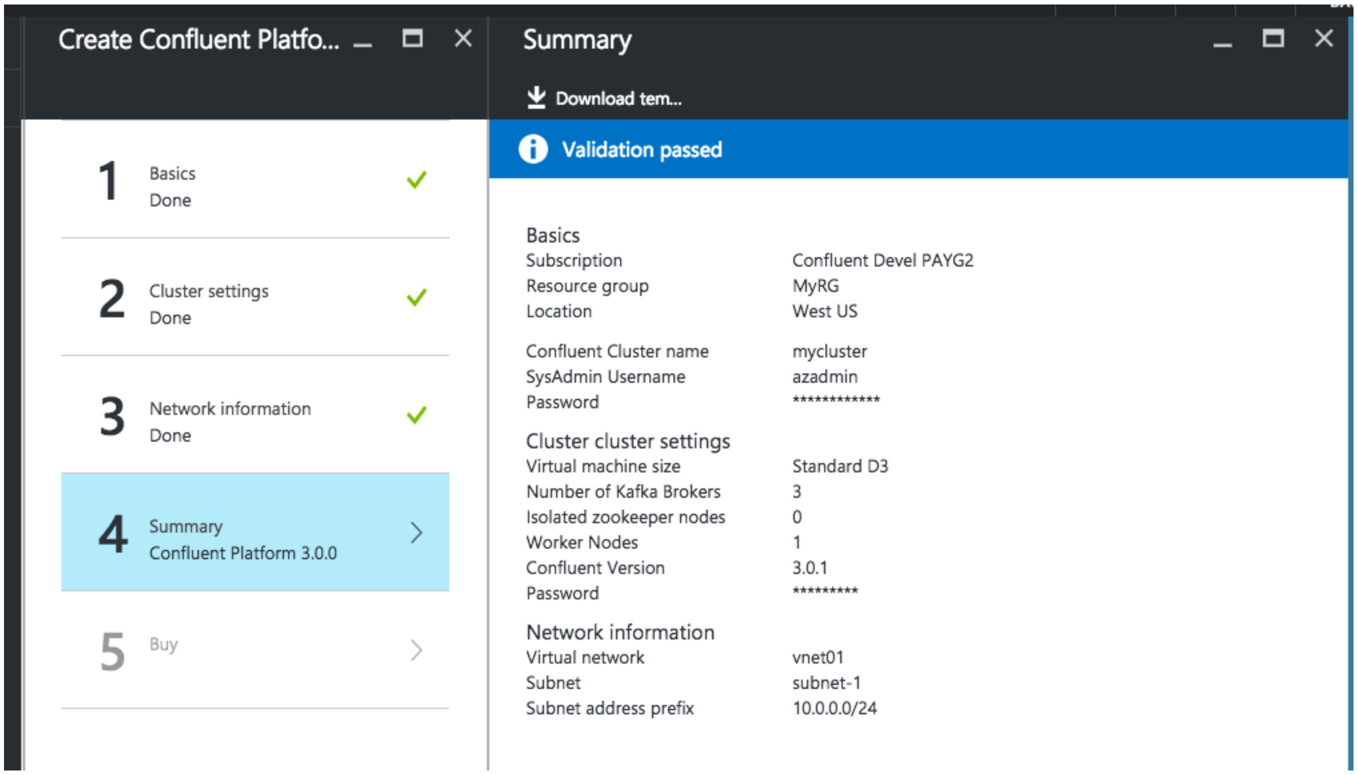
* Subnet name
subnet-1

* Subnet address prefix
10.0.0.0/24

When selecting a pre-existing network, you'll want to make sure that specified subnet does not restrict traffic on the ports necessary for the Kafka services.   This is highly unlikely, but worth a quick check of core Confluent Enterprise ports (2181, 2888, 3888, 8081, 8082, 8083, 9021, and 9092).

IV. Summary

A single-page overview of what you have entered will be displayed, you can return to the previous blades in the event you need to change any settings before starting the deployment.


Create Confluent Platfo...
1 Basics Done ✓
2 Cluster settings Done ✓
3 Network information Done ✓
4 Summary Confluent Platform 3.0.0
5 Buy
Summary
Download tem...
Validation passed

Basics

Subscription	Confluent Devel PAYG2
Resource group	MyRG
Location	West US
Confluent Cluster name	mycluster
SysAdmin Username	azadmin
Password	************

Cluster cluster settings

Virtual machine size	Standard D3
Number of Kafka Brokers	3
Isolated zookeeper nodes	0
Worker Nodes	1
Confluent Version	3.0.1
Password	*********

Network information

Virtual network	vnet01
Subnet	subnet-1
Subnet address prefix	10.0.0.0/24

V. Buy

Confirm the final deployment. Note that Confluent Enterprise on Azure is deployed in a bring-your-own-license model. There will be no charges incurred beyond the infrastructure costs. The enterprise-licensed features will be available in trial mode for 30 days following the deployment.

Provisioning the Azure infrastructure and configuring the Confluent software will take 10 to 20 minutes. You can track the progress of the deployment in the Azure Portal by monitoring the Resource Group you selected for the deployment.

Testing the deployment

There are several different ways to connect with the running Confluent Enterprise services on Azure. Whatever system you use for these tests must have proper network connectivity to the resources hosting the deployment. For simplicity, it is best to utilize a server in the same virtual network as the cluster. Servers outside the virtual network can access the hosts in Azure with proper settings in their hostname resolution framework. Details on that configuration are available below in Useful Azure Commands .

The following sections describe some simple options. We'll run these tests from any of the worker nodes, as those are guaranteed to have the proper network connectivity. You should connect to the worker node via ssh as user "kadmin". If you configured the deployment to use sshPublicKey access, the same key you deployed for the System Administrator will work for the kadmin user. Otherwise, you'll use the password you defined in the "Cluster settings" blade above. The Confluent software is installed in /opt/confluent; the kadmin user will have the proper PATH settings to invoke all of the necessary commands from that location.

Many of the commands require the proper specification of the zookeeper and broker nodes. The internal hostnames are of the form <ClusterName>-<group>-<n> (i.e. mycluster-broker-2, mycluster-zk-1, or mycluster-worker-1). Recall that the zookeeper quorum may be deployed on the broker nodes … in which case you'll use mycluster-broker-1 for the zookeeper host specification.

A. List / create Kafka topics

Kafka topic metadata operations are done via the kafka-topics command

```
$ kafka-topics  --zookeeper mycluster-zk-1:2181  --list
```

```
 "connect-configs",
 "connect-offsets",
 "connect-status",
 "_schemas",
 "__consumer_offsets"
```

In this case, you can see the topics created by the Schema Registry and the Kafka Connect services. The Confluent Control Center will also create a number of _confluent-controlcenter-* topics. You can go on to create a new topic with

```
$ kafka-topics --zookeeper mycluster-zk-1:2181 --create \
    --topic vtest1 --partitions 1 --replication-factor 3
```

```
 Created topic “vtest1”.
```

```
$ kafka-topics  --zookeeper mycluster-zk-1:2181  --list

  "connect-configs",
  "connect-offsets",
  "connect-status",
  "_schemas",
  "__consumer_offsets",
  “vtest1”
```

B. Producing / consuming messages via the REST Proxy interface

The Confluent REST Proxy service is available at all the broker nodes. That allows you to very simply send or receive messages from a given topic. Each of the broker nodes will host an instance of the REST Proxy server. This configuration will allow you to post messages to the newly created vtest1 topic as follows:

```
$ export RPURL=http://mycluster-worker-1:8082

$ curl -X POST -H "Content-Type: application/vnd.kafka.json.v1+json" \

   --data '{"records":[{"value":{"foo":"bar"}}]}' RPURL/topics/vtest1

{"offsets":[{"partition":0,"offset":0,"error_code":null,"error":null}],"key_schema_id":null,"value_sc
hema_id":null}
```

```
$ curl -X POST -H "Content-Type: application/vnd.kafka.json.v1+json" \

   --data '{"records":[{"value":{"foo":"baz"}}]}' $RPURL/topics/vtest1

{"offsets":[{"partition":0,"offset":1,"error_code":null,"error":null}],"key_schema_id":null,"value_sc
hema_id":null}
```

Those messages can be read back by creating a temporary consumer (which you should be careful to delete when you’re finished)

```
$ curl -X POST -H "Content-Type: application/vnd.kafka.v1+json" \
   --data '{"name": "ext_consumer_instance",
          "format": "json",   "auto.offset.reset": "smallest"}' \
  $RPURL/consumers/ext_json_consumer
```

```
{"instance_id":"ext_consumer_instance",
"base_uri":"http://mycluster-worker-1:8082/
consumers/ext_json_consumer/instances/ext_consumer_instance"}
```

```
$ curl -X GET -H "Accept: application/vnd.kafka.json.v1+json" \
$RPURL/consumers/ext_json_consumer/instances/ext_consumer_instance/topics/vtest1
```

```
[{"key":null,"value":{"foo":"bar"},"partition":0,"offset":0},{"key":null,"value":{"foo":"baz"},"partition"
:0,"offset":1}]
```

```
$ curl -X DELETE $RPURL/consumers/ext_json_consumer/instances/ext_consumer_instance
```

# No content in response

C. Deploying a connector

Your Confluent Enterprise deployment will have 1 or more nodes where the Kafka Connect worker containers are running.   Each node supports a REST interface that can be used to configure and run the available connectors. This configuration can be done directly from the cluster nodes or via the Confluent Control Center.   We'll discuss the direct option first.

You can query any worker for which connectors are available with a simple command.

```
$ export CWURL=http://mycluster-worker-1:8083
```

```
$ curl $CWURL/connector-plugins
```

```
[{"class":"io.confluent.connect.hdfs.HdfsSinkConnector"},
{"class":"io.confluent.connect.jdbc.JdbcSourceConnector"},
{"class":"org.apache.kafka.connect.file.FileStreamSourceConnector"},
{"class":"io.confluent.connect.hdfs.tools.SchemaSourceConnector"},
{"class":"org.apache.kafka.connect.file.FileStreamSinkConnector"}]
```

You can configure an available connector by uploading the appropriate configuration settings.

```
$ curl –X PUT  -H "Content-Type: application/json" \
    --data ‘{“name”: “SourceTest1”, “config”: {"connector.class":
 "org.apache.kafka.connect.file.FileStreamSourceConnector",
 "topic": "vtest1",
 "file": "/opt/confluent/etc/kafka/connect-distributed.properties"}}’\
 $CWURL/connectors
```

```
$ curl $CWURL/connectors
```

```
["SourceTest1”]
```

```
$ kafka-console-consumer --new-consumer \
   --bootstrap-server mycluster-broker-1:9092 \
   --from-beginning --topic vtest1 \
   --max-messages 10
```

```
$ # The first 10 lines of connect-distributed.properties will
$ # be displayed since the SourceConnector was configured to
$ # send each line from that file to the “vtest1” topic.
```

This example was for the simplest of source connectors, just dumping the contents of a file into the Kafka topic created above.   Other connectors will have more complex configurations.   Please see the documentation on Kafka Connectors for more details on building data pipelines in your Confluent Platform deployment.

D. Accessing the Control Center

Confluent Control Center is accessed via a web interface. By default, Control Center is deployed on worker node 1 of your deployment. Identify the public IP address of that node, and use your browser to connect to

http://<cluster>-<random>-worker-1.<azuredomain>:9021

The actual link will look something like

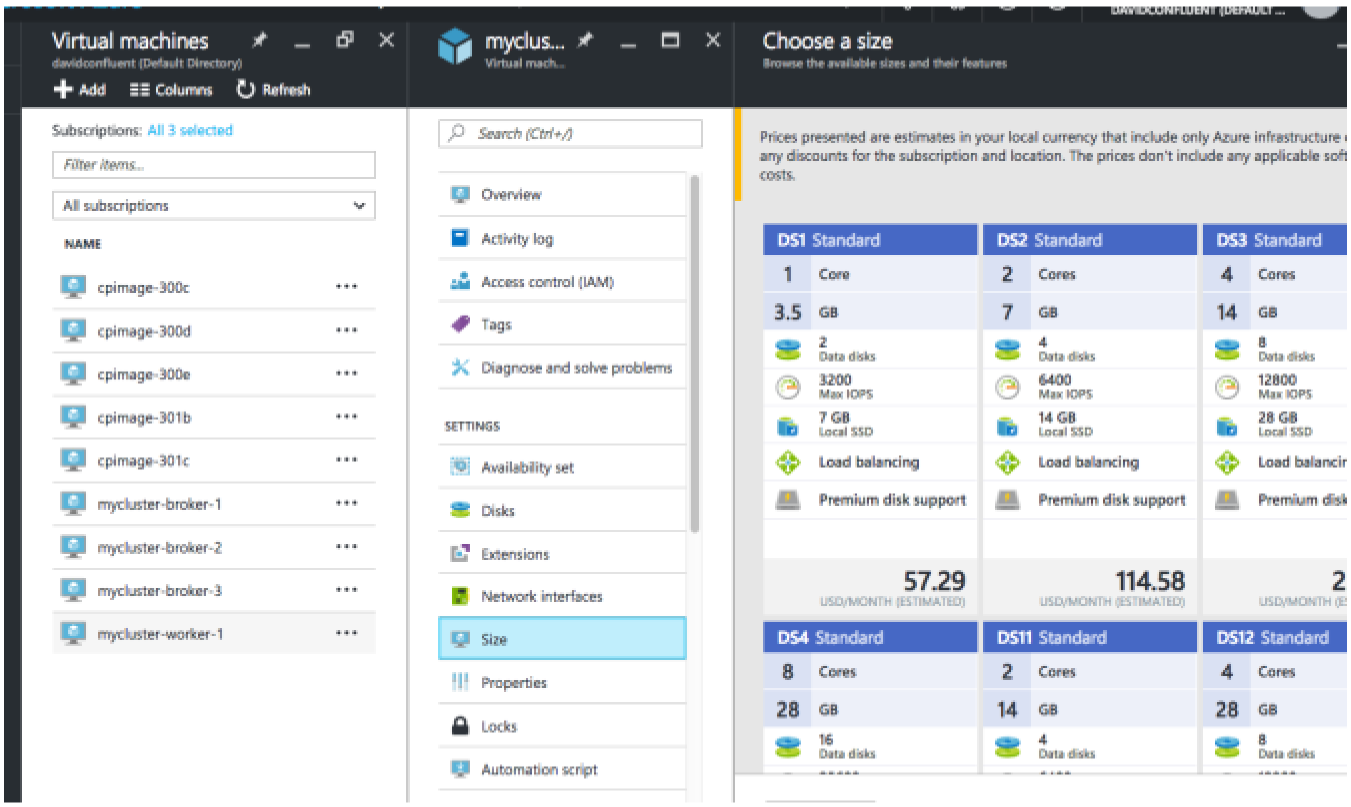
http://mycluster-gdbxxg64-worker-1.westus.cloudapp.azure.com:9021

From the Control Center, you can deploy instances of the available connectors and monitor the status of data streaming through the Kafka topics. Details on the Control Center can be found at http://docs.confluent.io/current/control-center/docs/index.html .

Updating the Deployment

All Confluent Enterprise services deployed within Azure are readily modifiable. Here are some common changes that you might wish to make.

A. Change the VM size of a running instance
   The Azure portal allows you to redeploy any of the instances in your Confluent topology to a different size should the need arise. For example, you may want to expand the memory capacity of the worker nodes to support a greater number of deployed Connectors. The screenshot below illustrates how to change the size for the worker node in our sample cluster (easily identified as <cluster>-worker-<n> in the list of Virtual Machines) :



Virtual machines
davidconfluent (Default Directory)
Add · Columns · Refresh

Subscriptions: All 3 selected
Filter items...
All subscriptions

NAME
cpimage-300c
cpimage-300d
cpimage-300e
cpimage-301b
cpimage-301c
mycluster-broker-1
mycluster-broker-2
mycluster-broker-3
mycluster-worker-1

Choose a size
Browse the available sizes and their features

Prices presented are estimates in your local currency that include only Azure infrastructure
any discounts for the subscription and location. The prices don't include any applicable soft
costs.

DS1 Standard	DS2 Standard	DS3 Standard
1 Core	2 Cores	4 Cores
3.5 GB	7 GB	14 GB
2 Data disks	4 Data disks	8 Data disks
3200 Max IOPS	6400 Max IOPS	12800 Max IOPS
7 GB Local SSD	14 GB Local SSD	28 GB Local SSD
Load balancing	Load balancing	Load balancir
Premium disk support	Premium disk support	Premium disk
57.29 USD/MONTH (ESTIMATED)	114.58 USD/MONTH (ESTIMATED)	2 USD/MONTH (E

DS4 Standard	DS11 Standard	DS12 Standard
8 Cores	2 Cores	4 Cores
28 GB	14 GB	28 GB
16 Data disks	4 Data disks	8 Data disks

When changing the size of your nodes, be sure that each resize operation is complete and the Confluent services on the node are back on line before attempting a resize operation on another node.

B. Make additional Kafka Connectors available on the worker nodes
   A wide range of Connectors is available for the Confluent deployment. Only the Connectors supported directly by Confluent are initially deployed into your Azure cluster, but you are free to

install your own connectors or those developed by the Kafka Connect community. To add them appropriately, you'll need to do the following steps on each of the worker nodes in your cluster:

a. Log on to the worker node as user kadmin
b. Create a directory for the connect jar files: /opt/confluent/share/java/kafka-connect-extras. The directory MUST be named kafka-connect-<new-connector> so that the worker tasks will properly load the java classes.
c. Copy the jar file for your connectors into the newly created directory
d. Restart the Kafka Connect worker process with the command
   sudo service cp-connect-service restart
e. Confirm the availability of the new connector with
   curl http://localhost:8083/connector-plugins | jq .

Take care to perform these steps on all the worker nodes in your cluster, as the Kafka Connect framework will automatically provision Connector tasks on any available worker node.

Useful Azure Commands

The Azure command line tool (available from https://docs.microsoft.com/en-us/cli/azure/install-azure-cli) can be used to track the infrastructure resources assigned to your Confluent deployment. In conjunction with other shell tools (including jq), the command can be very powerful. Here are some helpful examples.

A. List out the public IP addresses and hostnames of a cluster deployed in resource group MyRG

```
azure network public-ip list --json MyRG | \
   jq -r '.[] | "\(.ipAddress) \(.dnsSettings.domainNameLabel)"' | \
   awk '{split($2,a,"-"); print $1" "$2"  "a[1]"-"a[3]"-"a[4]}'
```

The 'awk' filter at the end of the command has the effect of stripping out the random seed string from the hostnames. That randomness is necessary for the public hostnams to avoid namespace collisions within cloudapp.azure.com, but unnecessary within the internal network. The resulting output :

```
13.91.58.24   confsm-zm5awowo-broker-1 confsm-broker-1
13.93.147.47  confsm-zm5awowo-broker-2 confsm-broker-2
13.91.62.4    confsm-zm5awowo-broker-3 confsm-broker-3
13.93.150.33  confsm-zm5awowo-worker-1 confsm-worker-1
13.93.147.183 confsm-gwpnqjw5-zk-1 confsm-zk-1
```

can be inserted directly in your local /etc/hosts file for easier access to the cluster with the standard Kafka clients.

Troubleshooting

The Azure Portal allows you to view the service- and instance-level details for all the components in the Confluent topology. This section outlines several common approaches to identifying and resolving configuration issues in your cluster.

Newly configured connector is not available

After updating your deployment to support an additional Connector, you are unable to confirm the availability of the connector with the

```
curl http://localhost:8083/connector-plugins | jq .
```

command. Some community connectors may contain Java classes that conflict with the connectors distributed as part of Confluent Enterprise. Error messages in the Kafka Connect log file (/opt/confluent/logs/connectDistributed.out) will likely exist in the event of such conflicts. A good practice is to deploy the new connector in standalone mode *before* adding it to the default worker CLASSPATH.

As discussed earlier, the Connect Worker automatically includes all the Java libraries in the /opt/confluent/share/java/kafka-connect-* directories. This enables you to isolate your connector by simply renaming those directories and then restarting the Connect Worker service.

Confluent Control Center is unresponsive

The Control Center application is constantly aggregating metrics from all the brokers and topics in your cluster. On rare occasions, the processing of this data can temporarily disrupt its response to web page input (mouse clicks and keystrokes). If the situation persists for more than a few minutes, you may need to increase the CPU and memory capacity of the worker-1 instance.