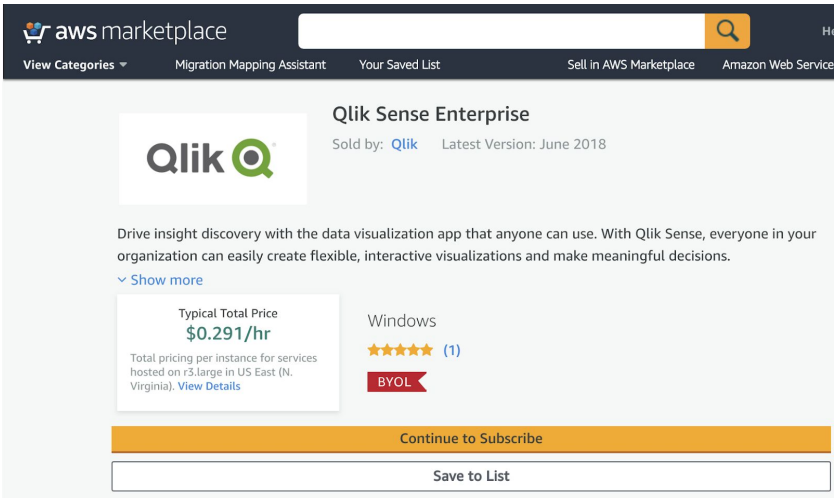
Installation of Qlik Sense Enterprise in the cloud using DevOps concepts

This article describes one technique for deploying Qlik Sense Enterprise for Windows in the cloud (Amazon AWS and Microsoft Azure) using Hashicorp Terraform to apply a declarative codified configuration. This configuration ensures the same deployment is performed each time it is executed, the configuration is easily added to change management technologies and documents exactly what is going to be provisioned. Deploying Qlik Sense Enterprise in this fashion allows for a known configuration to be applied consistently and reliably.

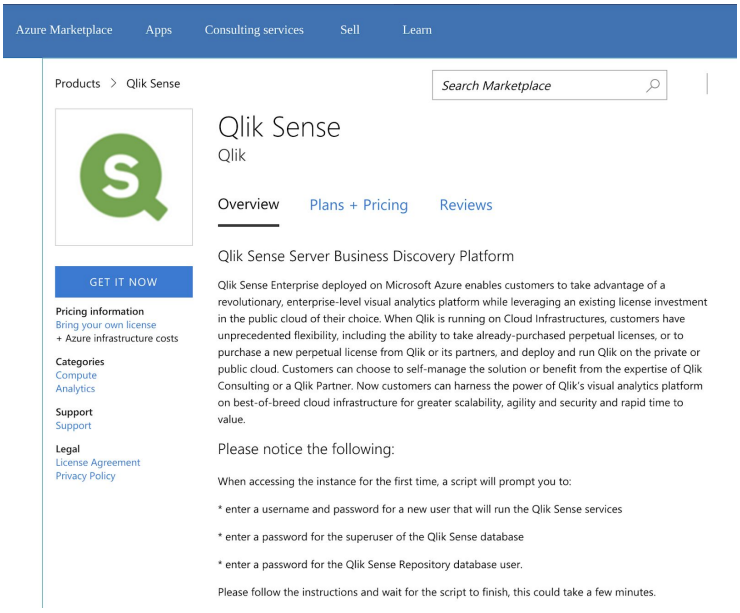
Cloud Marketplace

Both Amazon Web Services and Azure provide marketplaces for software to be available and maintained by vendors (like Qlik). The benefits of using these resources is they are updated by Qlik (when versions are released) and they are known entities.

https://aws.amazon.com/marketplace/pp/B01M5HCC0D?qid=1534115067308&sr=0-1&ref_=srh_res_product_title



https://azuremarketplace.microsoft.com/en-us/marketplace/apps/qlik.qlik-sense?tab=Overview

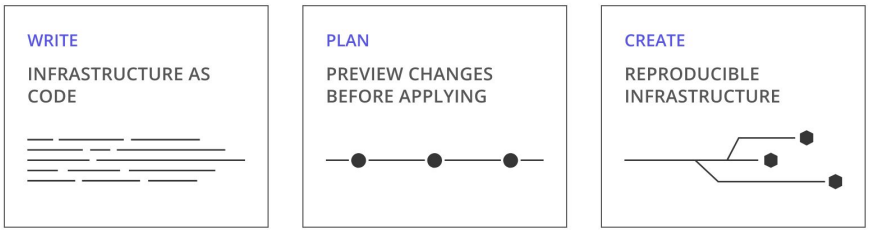


Both of these services enable you to create a machine running the latest version of Qlik Sense Enterprise quickly and efficiently within the respective cloud platform (AWS or Azure).

Introducing Terraform

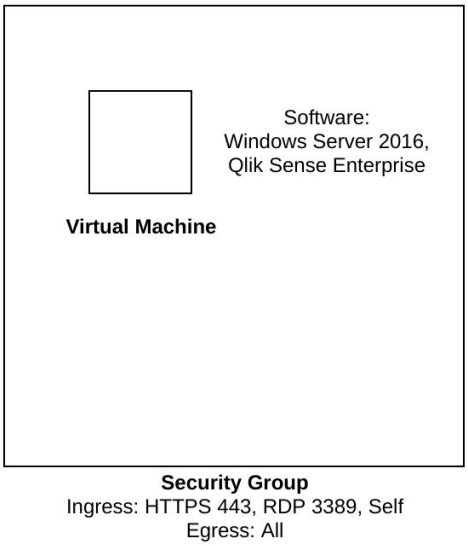
Terraform is an open source tool that enables you to create infrastructure (machines and applications) based on configuration files in a safe and predictable fashion. Configuration files describe the components required to run an application, Terraform takes this configuration file, generates a plan that describes what will happen then executes the plan to build the desired infrastructure.

Using Terraform infrastructure is described as code (Infrastructure as code) which allows a blueprint of your application to be versioned and stored as any other code would be. This can then be shared and re-used.



AWS EC2 CONFIGURATION

A sample Terraform configuration is available on the following URL AWS EC2 Terraform configuration. The configuration files are declarative, they define exactly what needs to be created to build the desired infrastructure. The files have been defined into separate TF files that create the various components of the infrastructure. Running this configuration will create the following infrastructure:



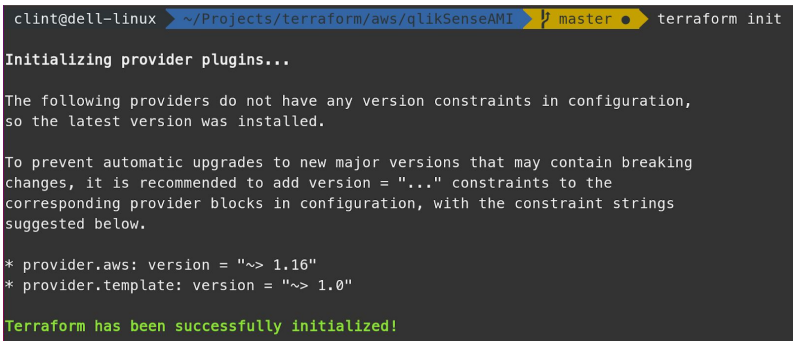
Prerequisites

1. An Amazon EC2 account
2. You must subscribe to the Qlik Sense AMI (https://aws.amazon.com/marketplace/pp/B01M5HCC0D?qid=1532912898605&sr=0-2&ref_=srh_res_product_title) - This only needs to be done once on the account
3. You need an EC2 access_key and security_key (Get this by creating an IAM within your AWS account)
4. Qlik Sense license (the Qlik Sense Enterprise AMI is Bring your own License)
5. Install Terraform (https://www.terraform.io/downloads.html)
6. Download or clone the repository (AWS EC2 Terraform configuration)

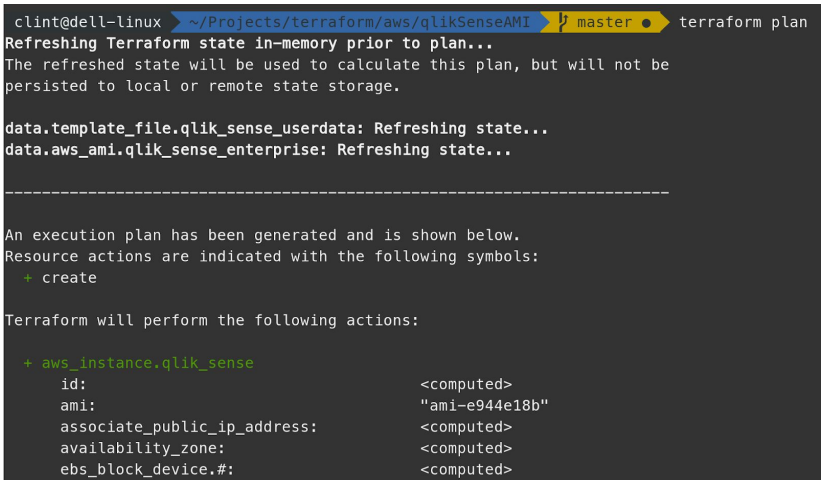
Deployment

With the prerequisites all complete deploying Qlik Sense Enterprise is a simple command away. But before we do that we need to initialise Terraform, this will download any components that the configuration requires.

1. Run *terraform init* from your shell (PowerShell, Bash)



2. Run *terraform plan* this will execute the configuration and show you what will be created. This is one of the big advantages of using Terraform, you can see what is going to be created without actually creating it, you will see what will change if changes are made to the configuration allowing for a known state in the event of change management.

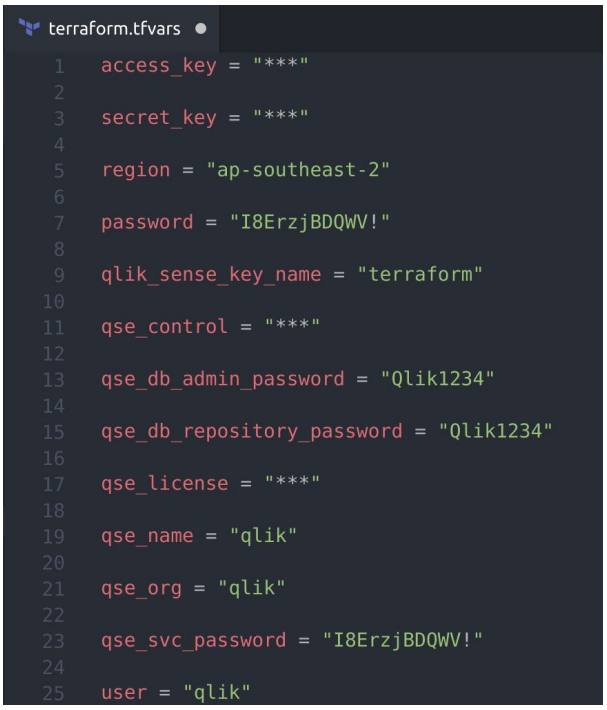


3. If the plan is satisfactory run *terraform apply* this will go ahead and apply the configuration after you provide the variables required.

Variable name	Variable purpose
access_key	IAM access key with API access to the AWS account
security_key	IAM security key associated with access_key
region	AWS region to create the infrastructure

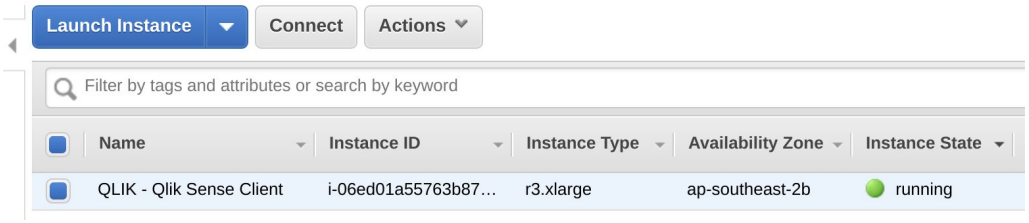
user	Windows user that will be created as an administrator
password	Windows password for user
qlik_sense_key_name	AWS Key pair (this is used to be able to decrypt the base Windows Administrator password)
qse_control	Qlik Sense Enterprise control number
qse_db_admin_password	Qlik Sense Enterprise password for db admin (Postgres)
qse_license	Qlik Sense Enterprise serial number
qse_name	Qlik Sense Enterprise name of license owner
qse_org	Qlik Sense Enterprise organisation of license owner
qse_svc_password	Qlik Sense Enterprise service account password

All of these variables can be added to a file with a tfvars extension (for example qlikSense.tfvars). Then the *terraform apply* will not prompt for values.



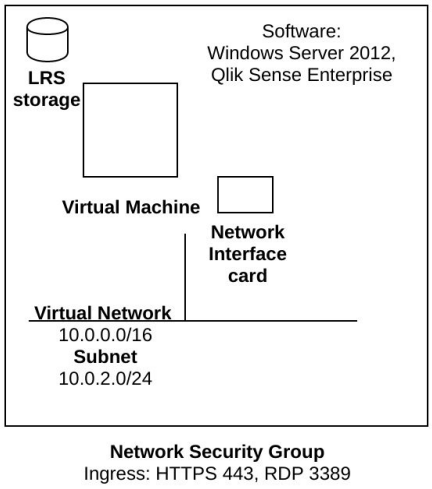
```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_security_group.qlik_sense_sg: Creating...
  arn:                                  "" => "<computed>"
  description:                          "" => "Access rules for the Qlik Sense"
  egress.#:                             "" => "1"
  egress.482069346.cidr_blocks.#:       "" => "1"
  egress.482069346.cidr_blocks.0:       "" => "0.0.0.0/0"
  egress.482069346.description:         "" => ""
  egress.482069346.from_port:           "" => "0"
  egress.482069346.ipv6_cidr_blocks.#:  "" => "0"
  egress.482069346.prefix_list_ids.#:   "" => "0"
```

```
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

Qlik Sense Public IP = 13.54.96.182
```



Azure configuration

A sample Terraform configuration is available on the following URL Azure Terraform configuration. The configuration files are declarative, they define exactly what needs to be created to build the desired infrastructure. The files have been defined into separate TF files that create the various components of the infrastructure. Running this configuration will create the following infrastructure (items in **bold**):



Prerequisites

1. A Microsoft Azure account
2. An identity created in Azure Active Directory for Terraform to be able to use the Azure APIs (https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-create-service-principal-portal)
3. Qlik Sense license
4. Accept the Qlik Sense Enterprise license terms via the Azure Marketplace
5. Qlik Sense license (the Qlik Sense Enterprise AMI is Bring your own License)
6. Install Terraform (https://www.terraform.io/downloads.html)
7. Download or clone the repository (Azure Terraform configuration)

Deployment

With the prerequisites all complete deploying Qlik Sense Enterprise is a simple command away. But before we do that we need to initialise Terraform, this will download any components that the configuration requires.

2. Run *terraform init* from your shell (PowerShell, Bash)

```
clint@dell-linux ~/Projects/terraform/azure master terraform init

Initializing provider plugins...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.azurerm: version = "~> 1.1"
* provider.template: version = "~> 1.0"

Terraform has been successfully initialized!
```

3. Run *terraform plan* this will execute the configuration and show you what will be created. This is one of the big advantages of using Terraform, you can see what is going to be created without actually creating it, you will see what will change if changes are made to the configuration allowing for a known state in the event of change management.

```
clint@dell-linux ~/Projects/terraform/azure master terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.

data.template_file.qlikSense_powershell_script: Refreshing state...

------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  + azurerm_network_interface.qlikSense_nic
      id:                                                    <computed>
      applied_dns_servers.#:                                 <computed>
      dns_servers.#:                                         <computed>
```

4. If the plan is satisfactory run *terraform apply* this will go ahead and apply the configuration after you provide the variables required.

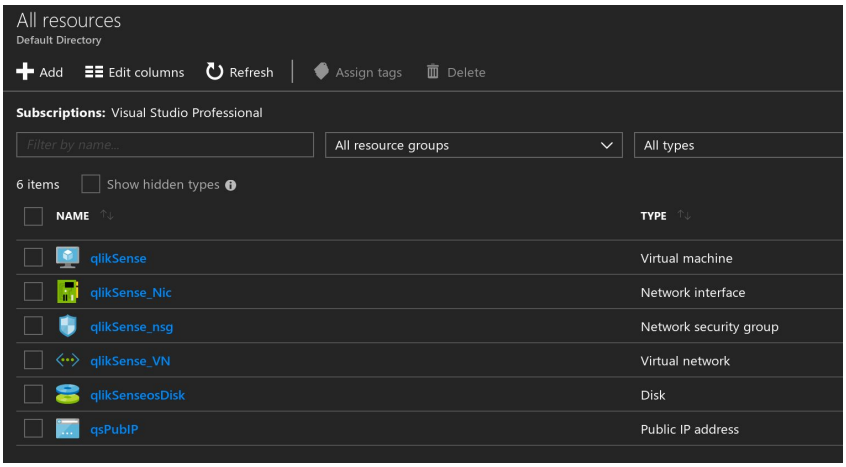
Variable name	Variable purpose
tenant_id	Azure Active Directory tenant id
client_id	Azure Active Directory client id
client_secret	Azure Active Directory client secret
subscription_id	Azure subscription id
qlikSense_rg	Azure resource group to create the infrastructure within
azure_location	Location to create the infrastructure
windows_administrator	Windows user that will be created as an administrator

windows_administrator_pass	Windows password for user
server_hostname	Server's computername
qse_control	Qlik Sense Enterprise control number
qse_db_admin_password	Qlik Sense Enterprise password for db admin (Postgres)
qse_license	Qlik Sense Enterprise serial number
qse_name	Qlik Sense Enterprise name of license owner
qse_org	Qlik Sense Enterprise organisation of license owner
qse_svc_password	Qlik Sense Enterprise service account password

All of these variables can be added to a file with a tfvars extension (for example qlikSense.tfvars). Then the *terraform apply* will not prompt for values.

```
Plan: 10 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

azurerm_resource_group.qlikSense: Creating...
  location: "" => "eastus"
  name:     "" => "qlikSense"
  tags.%:   "" => "<computed>"
```

```
-a610-...hines/qlikSense/extensions/install_qse)

Apply complete! Resources: 10 added, 0 changed, 0 destroyed.
```



Destroying the infrastructure

Destroying what has been built is performed by executing *terraform destroy*. Everything will be removed.

```
Destroy complete! Resources: 10 destroyed.
```