Create an AWS Lab Environment for Demos

Deploy Domain and Active Directory

* Create Cloud Formation stack from AWS Quickstart
* <https://docs.aws.amazon.com/quickstart/latest/active-directory-ds/step2.html>
* Deploy stack under scenario 2
* Test ability to connect to remote desktop gateway
* Modify the *DomainMembers* security group to allow 1433 TCP access on domain

Deploy Windows 2016 AMIs to Domain

* T2.Small or T2.Medium (SQL Server needs memory!)
* Deploy 2 AMI images to private or public subnet 1 from cloud formation stack
* Deploy 1 AMI image to private or public subnet 2 from cloud formation stack
* Add AMI images to the *DomainMembers* security group created by cloud formation stack
* If private connect to images by RDP via the remote desktop gateway already in the domain
* Install SQL Server 2017 Developer Edition
* <https://www.microsoft.com/en-us/sql-server/sql-server-downloads>
* Add AMIs to the domain you specified while building your cloud formation stack
* Change server names, customize to your preferences etc.

Deploy AWS Linux AMI (version 2 min) to Domain

*You can choose another linux distro if you are more comfortable with Ubuntu or Redhat etc.*

* T2.Small or T2.Medium
* Deploy 2 AMI images to public subnet 2 (makes SQL deploy easier from public subnet)
* Add the AMI images to the *DomainMembers* security group created by cloud formation stack
* Install and configure SQL Server using yum
* <https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-overview?view=sql-server-2017>
* Join Linux boxes to your domain
* <https://docs.aws.amazon.com/directoryservice/latest/admin-guide/join_linux_instance.html?icmpid=docs_dirservices_console>

Reduce AWS costs by keeping instances stopped when not actively working with the lab. In my scenario my total AWS costs for the lab, time spent building, creating demos etc, was less than 50$ total. If you are very prudent I could imagine a lab for less than 15$ monthly.

There are guides out on the internet for building labs using both VMWare and Azure as well. I specifically chose AWS simply for the ease of the pre-built AWS Cloud Formation stacks to build out my test domain, with domain controllers, internet access, and multi-subnet environment.

Note from the field:

I recommend turning termination protection on until you are done with the lab, I accidentally deleted one of my domain controllers and wasted a lot of time and effort.

Microsoft SQL Server AlwaysOn Availability Groups Checklist

*Node = each individual windows server participating in the cluster  
Replica = each individual sql server instance participating in the group  
Database = each individual database participating in the group*

Configure AlwaysOn *(Operating System)*

* Enable windows clustering on each node
  + Server Manager – Enable Failover Cluster Feature
* Add participating nodes to cluster
  + Failover cluster manager – create cluster
* Validate cluster
  + Validate all steps
  + Disk warnings related to shared disks are ok, AG doesn’t need shared disks
  + Verify redundant networks that validation can’t identify
* Create cluster
  + Create cluster
  + Configure quorum per best practices
    - No vote by default
    - Include all primaries
    - Include failover target nodes
    - Exclude secondary / asynchronous sites
    - Ensure odd number of voting nodes
    - Re-assess post failover scenarios
  + Establish file share witness if necessary
* Multi-subnet?
  + Firewall requirements
    - UDP 3343 – cluster service
    - TCP 3343 – node joining
    - TCP 135 – RPC calls between nodes
    - UDP 137 – Cluster administration
    - UDP 1024-65535 – Randomly allocated ports
    - UDP 49152-65535 – Randomly allocated ports
    - TCP 139/445 – File share witness
    - UDP 137/138 – File share witness and SMB / netbios
    - TCP 1433 – SQL Server
    - TCP 7022 or custom port – Mirror endpoint needed for AG
* Review event logs and c:\windows\cluster\reports logs
  + Dump cluster event log with PS Install-Module FailoverClusters Get-ClusterLog

Optional: configure AlwaysOn failover cluster with no domain

* DNS server with shared DNS suffix
* Local administrator with same name and same password
* Registry setting: LocalAccountTokenFilterPolicy –Value 1
* Create cluster as above
* <https://bit.ly/2NTq8O3> - *Microsoft documentation with additional information*

Configure Availability Groups *(SQL Server)*

* Enable AG on each SQL Server Windows instance
  + Windows configuration is done via the Configuration Manager
  + Linux configuration is done via the following commands

sudo /opt/mssql/bin/mssql-conf set hadr.hadrenabled 1

sudo systemctl restart mssql-server

* Create mirroring endpoint on each replica
  + SQL endpoints on Linux require certificate <https://bit.ly/2Dk9mDJ> (optional on Win)
* Grant connect on each endpoint/replica or specify certificate authentication in endpoint
* Set endpoint state to started
* Create an availability group (primary replica)
* Join secondary replicas to the availability group
* Join each secondary database to the new availability group (secondary replicas)

Configure Databases *(Database)*

* Join database to AG
* Configure synchronous / asynchronous replication
* Configure manual / automatic failover
* Configure read only / non-read only secondary

Configure a Listener *(Availability Group)*

* Select IP for each subnet
* Configure listener
* Test listener
* Test failovers

Configure Advanced Options

* Read only routing urls and lists
* Offloading backups – backup priority
* Failover behavior – flexible failover policy
* Setup monitoring

Full restrictions, limitations, recommendations, and best practices:

<https://docs.microsoft.com/en-us/sql/database-engine/availability-groups/windows/prereqs-restrictions-recommendations-always-on-availability?view=sql-server-2017>