**Section 5: Cryptogen tool**

**Cryptogen Tool**

**Cryptogen Binary:**

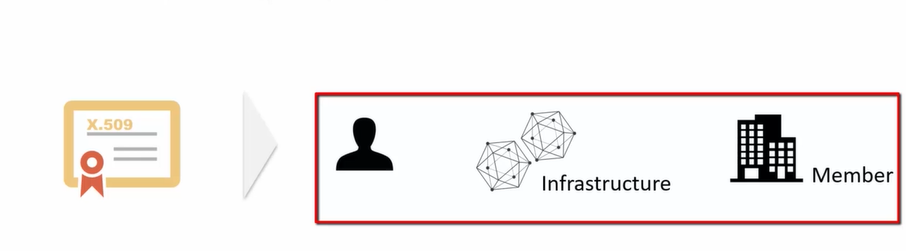
1. explain the usage of the crypto tool and
2. you should be able to generate some sample crypto material using the crypto tool.

The intent of this lecture is not to describe the configuration file that is used by the crypto gen tool.

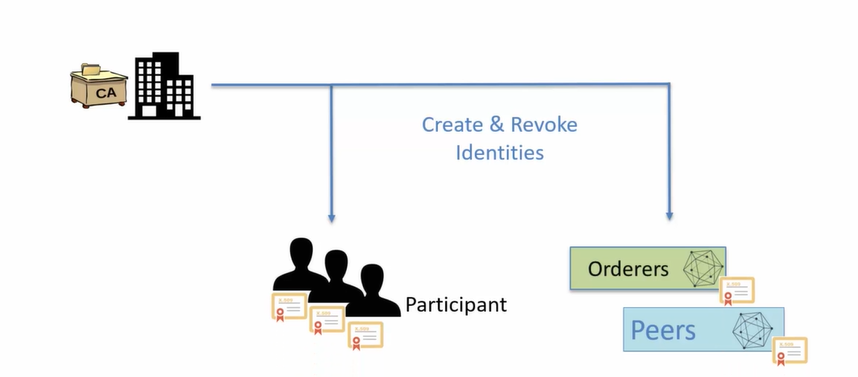
It is just to show you the various options and commands available on the crypto gen utility.

**let's recap how Permissioning is implemented on Hyperledger Fabric.**

* Hyperledger Fabric uses public key infrastructure for Permissioning certificates are issued to all the identities within the network.
* Now identities refer to not just the participants. It also refers to the infrastructure components such **as Orderers**, such as **peers**
* identities are also issued to the members or organizations that are part of the network.
* In other words, a certificate is issued to each of these identity types.



* The certificates are managed by the certification authorities, so each of the participants and the network components such as orderers and peers are issued a certificate by the certification authority.
* certificates are issued by the CA to ensure trust, verify identities, enforce access control, and secure the communication among participants and network components

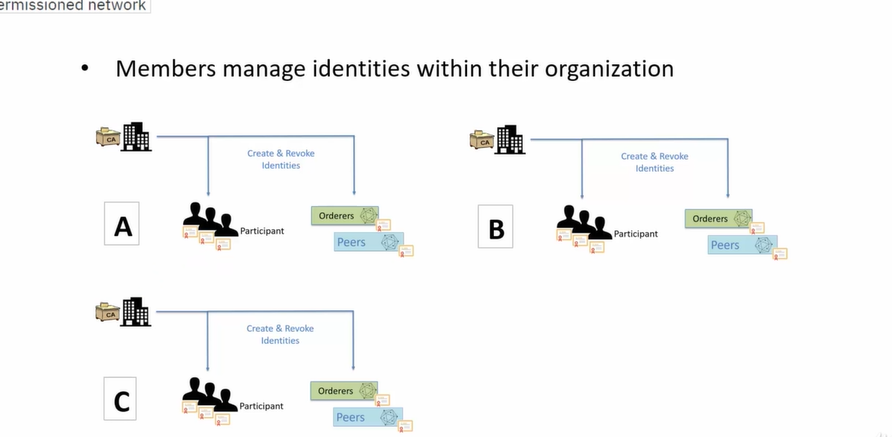


* Now, one important thing to understand here is that the identity management is not centralized. In other words, there is no single entity that is responsible for issuing the certificates.

Each member organization manages the identities within their organization.

So if a network has, let's say, three members, member A, member, B, member C,

each of these members will have their own separate certification authority, which will manage the certificates issued to the participants as well as to the network components such as orders and peers.



**Cryptogen Tool:**

**“The Cryptogen tool is a command-line utility in Hyperledger Fabric used to generate cryptographic material, such as certificates and keys, for network participants in a Fabric blockchain network. This tool is particularly helpful during development and testing to quickly generate the necessary credentials for identities like peers, orderers, and clients without needing a full-fledged Certificate Authority (CA).”**

* Cryptogen is a utility to generate cryptographic identities and certificates quickly for use in a Hyperledger Fabric network, particularly in development environments.
* It uses the crypto-config.yaml file for configuration.
* It creates **MSP** directories for each organization in the network, containing keys, certificates, and other necessary cryptographic material.
* **MSP directories are secure folders** for each organization in a blockchain network, containing cryptographic materials like keys, certificates, and root CAs. These directories manage identities, enable access control, and secure communication, ensuring that each organization can be trusted and authenticated within the network.

crypto tool is a command line utility for generating the crypto material.

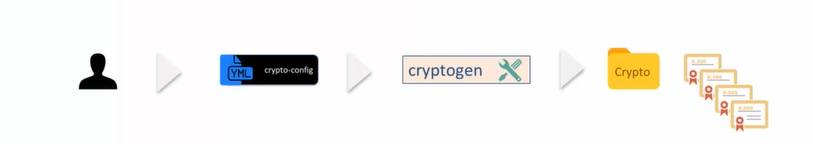
Crypto material refers to the certificate and the Keystore.

The crypto material generated by the crypto tool is typically used for setting up the test environments.

It requires the developer to provide the configuration information in Yaml format.

**At a high level The flow looks like this:**

1. **Creating the Crypto Config YAML**: As a developer, you first create a configuration file called crypto-config.yaml. This file defines the details of each participant organization and infrastructure component in the network, including orderers (responsible for transaction ordering) and peers (responsible for transaction validation and storage).
2. **Using the CryptoGen Tool**: You then launch a tool called **CryptoGen**, providing it with the crypto-config.yaml file. CryptoGen is a utility that reads the configuration details in the YAML file.
3. **Generating Cryptographic Material**: CryptoGen processes the configuration and generates all necessary cryptographic material, such as private keys, certificates, and identity information, based on the YAML file specifications.
4. **Output Folder with Certificates and Key Stores**: After CryptoGen completes, it saves all generated keys and certificates in a local folder. This folder includes key stores for all identities specified in the YAML file, creating all the necessary security credentials for each entity in the network.



The important point to note about crypto gen tool is that it is not meant to be used in production because real identities are issued by certification authorities managed at the organization or the member level.

Cryptogen tool is a **command line tool.**

In other words, you need to execute the crypto gen binary on the command shell or the command prompt.

**In Windows->**It can be the PowerShell or the command shell

**Unix and Mac OS->** it can be a terminal window.

**The general format of the command is :**



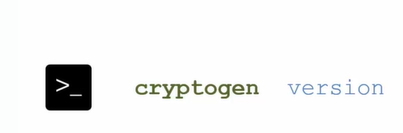
1. With crypto gen help you can get the help on all the commands which are available



1. for crypto gen to get help on specific command,



1. can be used to check the version of the crypto binary



1. show template will show you the template for the cryptogen configuration file. This is the Yaml format that I was referring to.



You can save the sample template in a file and then use it for creating your own environment specific configuration.

The cryptogen showtemplate command displays a **template for the cryptogen configuration file** in YAML format. This template shows the structure and necessary fields for defining the participants (organizations, peers, orderers, etc.) in the network.

The template is essentially a guide to help you set up the crypto-config.yaml file, where you specify details for each organization and component.

**Let’s see these commands in action.**

Let's start with the **Help** command.

The Help command shows all the commands that are available on the crypto gen binary.

Earlier I mentioned that you can get help on specific command by using the command name, let's say

**show template --help**

So this will give you the help on a specific command, which in this case is show template.

In fact, there is another way of getting help for a command, and that is to use **cryptogen help command name** in this case show template-> **cryptogen help showtemplate**

It is the same as the previous command

to get the version of the crypto gen binary->**cryptogen version.**

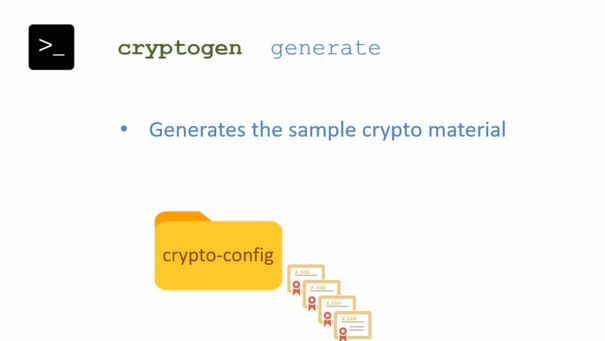
**cryptogen show template->**This will dump the sample configuration.For the crypto tool on the terminal.Obviously it's not very useful just by dumping it on the terminal.You can in fact save this template by piping the output to a file.

**cryptogen show template > temp.yaml 🡪**simply piping the output to a file and, I new file is created here temp**.**yaml and if I cat it it's the contents so I can go ahead and edit this file to my needs.

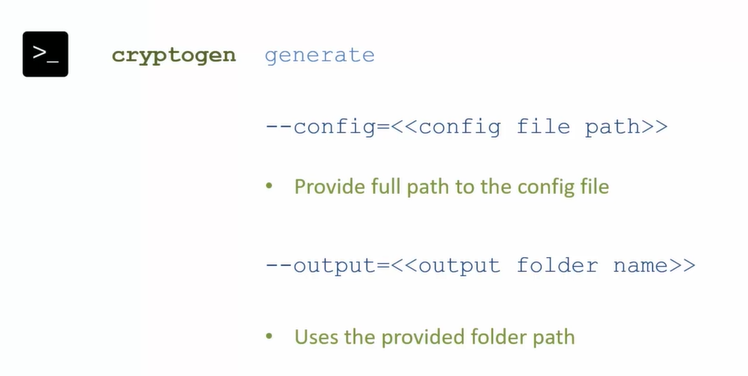
**Cat temp.yaml ->**display the contents of the temp.yaml file in the terminal. The cat command reads and outputs the entire file, allowing you to quickly view its content without opening it in an editor.

| **Command** | **Description** |
| --- | --- |
| **cryptogen --help** | **Displays all available commands for the cryptogen binary.** |
| **cryptogen showtemplate --help** | **Shows help documentation specifically for the showtemplate command.** |
| **cryptogen help showtemplate** | **Alternative way to get help on the showtemplate command (same as the previous).** |
| **cryptogen version** | **Displays the version of the cryptogen binary.** |
| **cryptogen showtemplate** | **Dumps a sample configuration template for the crypto tool directly in the terminal.** |
| **cryptogen showtemplate > temp.yaml** | **Pipes the template output to a new file temp.yaml.** |
| **cat temp.yaml** | **Displays the contents of temp.yaml in the terminal.** |
| **Cryptogen generate** | to generate a sample crypto material folder and the sample crypto material is generated under a folder by the name crypto dash config. |

You can execute the command **cryptogen generate** to generate a sample crypto material folder and the sample crypto material is generated under a folder by the name crypto dash config.



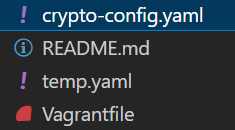
* The configuration used for generating this crypto material is the same configuration that is generated by way of the command crypto gen show template.
* So there are two organizations and one order that is predefined in this sample template.
* Now this command is not very useful.
* There are two flags that you can use with this command.



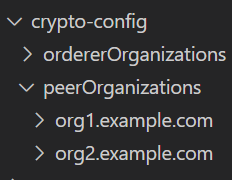
1. allows you to provide the full path to a config file
2. and you can also define the path to the output folder. So instead of using the default crypto -config as the folder name, you can provide a full folder path that will be used by the command for generating the crypto material

To demonstrate the use of the **generate** command:

1. **Generate the Sample Config**: First, a sample configuration is created by piping the template output to a file named crypto-config.yaml. This YAML file is the configuration file that defines the organizations, peers, and other participants for which cryptographic material will be generated.



1. **Run the Generate Command**: Next, the cryptogen generate command is run, with the --config flag pointing to the path of the crypto-config.yaml file. This tells cryptogen to use this specific configuration file as the input for generating cryptographic material.
2. **Specify Output Folder (Optional)**: By default, cryptogen will output the generated cryptographic material to a folder named crypto-config. Although it's possible to specify a custom folder using the --output flag, the default folder name is used in this demo.
3. **Generated Crypto Material**: Once the command completes, it creates a folder called crypto-config. Inside this folder, you’ll find all the necessary cryptographic materials (such as certificates and keys) for each entity (organizations, peers, and orderers) specified in the configuration file.
4. **Sample Organizations**: Since this is a sample configuration file, the generated output includes example organizations. Under the “peer org” section, there are two example organizations, each with its own cryptographic material, representing entities in the sample network.



Commands to run on terminal:

1. cryptogen showtemplate > crypto-config.yaml
2. cryptogen generate --config=./crypto-config.yaml
3. org1.example.com
4. org2.example.com

The cryptogen generate command(**used to generate cryptographic material such as certificates, keys, and identity credentials for organizations**) takes the crypto-config.yaml file as input, which defines **which organizations and peers** need cryptographic material. Upon execution, the command generates all the necessary **keys and certificates** in a default folder named crypto-config/. This is useful for blockchain networks like Hyperledger, where these certificates are essential for **identity management and secure communication**.

**Summary**

* Crypto Gen Tool is a command line utility for generating crypto material for testing.
* **cryptogen help** to get the list of commands supported by the crypto gen tool
* **cryptogen command help** To get help on a specific command
* **cryptogen showtemplate** show you the template that you can use for creating your own configuration file.
* The user has to provide this configuration file as an input to the **cryptogen Generate** command to generate the crypto material.

One important thing to keep in mind is that this tool is used for generating the crypto material fortesting. So in other words, you will not use this tool for generating the crypto material for production.



**Cryptogen tool configuration setup**

* **Orderer**: The orderer is responsible for receiving transactions from peers and organizing them into blocks. It ensures that the transactions are ordered and consistent across the network.
* **Peer**: A peer is a node that maintains a copy of the blockchain and validates transactions. Peers can also endorse transactions, which is essential for consensus in the network.
* **Peer**
  + A peer is a node in the network that hosts and maintains a copy of the blockchain ledger and runs chaincode (smart contracts).
  + Peers validate and endorse transactions, ensuring they meet the network's policy before they can be committed.
  + There are different types of peers:
    - **Endorsing Peers**: These validate and simulate transactions based on chaincode logic.
    - **Committing Peers**: These receive the ordered transactions from the orderer and commit them to the ledger (A **ledger** in Hyperledger Fabric is an immutable record of all transactions in the network, maintained by each peer and consisting of a blockchain for transaction history and a world state for current data values).
  + Each organization typically operates one or more peers.
* **Orderer**
  + The orderer (or ordering service) is responsible for ensuring the correct order of transactions across the network.
  + It receives endorsed transactions from peers, orders them, and groups them into blocks.
  + These blocks are then distributed to all peers for validation and commitment to the ledger.
  + The ordering service is crucial for maintaining the consistency and integrity of the blockchain, as it establishes a single source of truth for transaction order.

**Here we will learn how to set up the Yaml configuration file for the Cryptogenic tool.**

The Cryptogenic tool is for test crypto material setup.

Why would you need to set up the test crypto material?

* You may use it for learning Hyperledger fabric.
* You may want to experiment with different configurations, so you will need the crypto material.
* Generating that same crypto material with fabric utilities is not as straightforward, and you may wantto set up your own development environment with specific configuration.

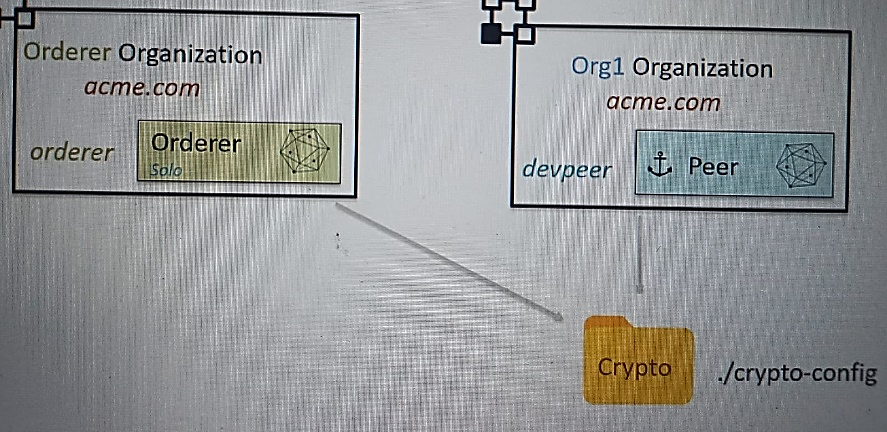
**Suggestion:**on setting up of the crypto config files before you set up the file, draw a diagram,put together the

* organizations
* orders and
* peers

that will be part of the configuration.

**sample setup:**

* There will be one organization with the name order in the domain acme.com
* there will be a org 1 organization within the same domain Acme.com
* In the order organization, there will be one order for which the host name is order and the order type is solo.
* And in the org 1 organization there is one peer with the host name Dev peer
* the name of the folder in which the crypto config material will be created is crypto -config.

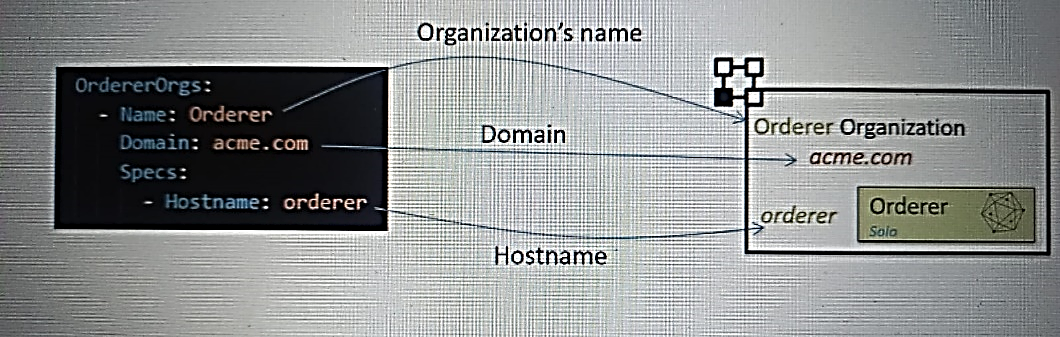


The crypto config yaml file has two sections

* order orgs-> which has a list of organizations managing the orders.
* peer orgs-> which is a list of member organizations that are managing peers.

**ORDER ORGANIZATION->**

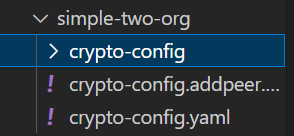
* You have the name of the order,
* then there is a domain definition for the order and then
* there is a specification for the order The host name. For example, here is order and you can have multiple such orders defined under the order orgs as it's a list.



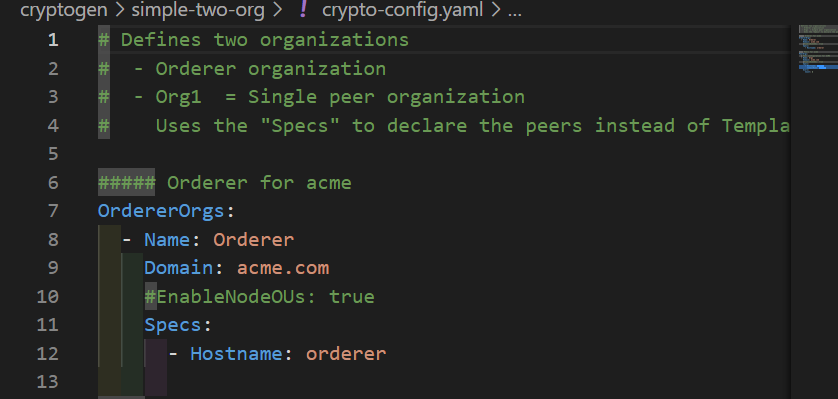
All commands in the demos in this lecture are getting executed under the **simple -tw-org** subfolder under **cryptogen** folder

crypto --config.yaml file.So ->just copy the content of this file to the crypto dash config dot yaml file under the simple-two-org**.**

**( Setup --> config --> simple\_two\_org)**

So this is the crypto dash config dot yaml file.



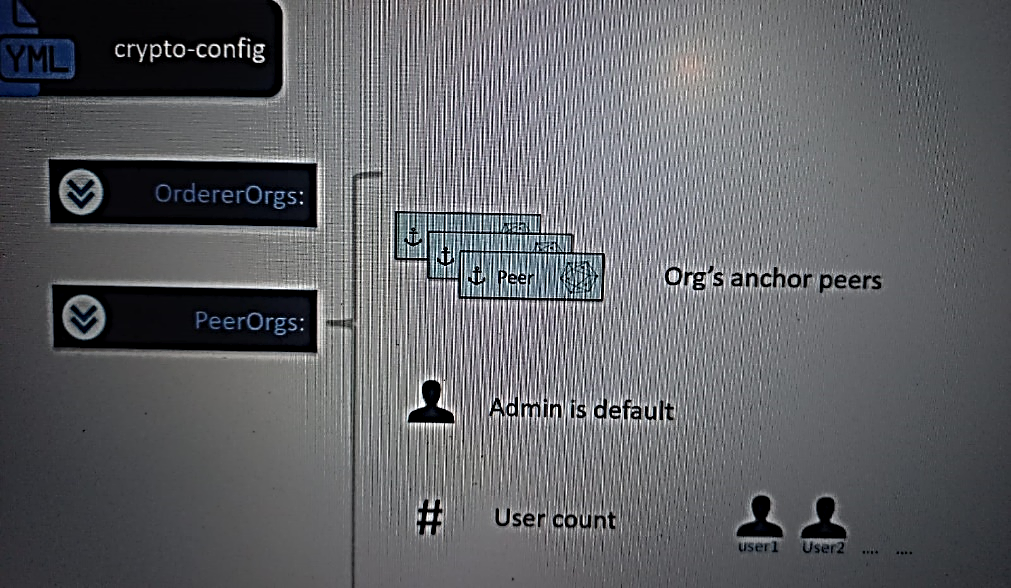
This is the order orgs section and we're going to execute

Vagrant->cryptogen->simple-two-orgs

**Command->**cryptogen generate --config=./crypto-config.yaml

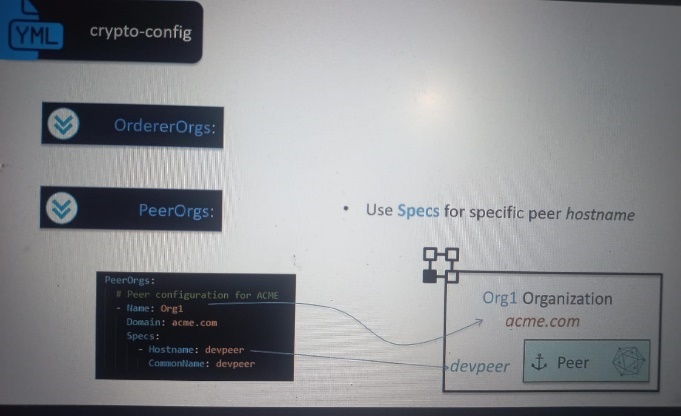
* **cryptogen**: This is the tool used to create cryptographic assets, like identities and certificates, for blockchain network components (e.g., peers, orderers).
* **generate**: This is a subcommand of cryptogen that tells it to generate the cryptographic material specified in the configuration file.
* **--config=./crypto-config.yaml**: This option specifies the path to the configuration file (crypto-config.yaml). The crypto-config.yaml file contains details about the organizations, orderers, and peers in the network. Here:
* **--config=** is the flag that tells cryptogen to use a configuration file.
* **./crypto-config.yaml** specifies the file path to crypto-config.yaml, where ./ refers to the current directory.
* Once this is executed, the crypto- config folder gets created and in this you will find that there is only **orders organizations**.
* There is no peer organization because at this time the file does not have the peer orgs section and
* within the order organization there is **one domain acme.com** under which there is one orderer under the orderers order is named order.acme.com.

**PEER ORGANIZATION->**



Within the peer orgs section,

1. **Creating a List of Organizations**: Within the "peer orgs" section, you’ll start by listing out the different organizations (orgs) that will participate in the network. Each organization typically represents a unique entity or business that interacts within the blockchain network.
2. **Defining Anchor Peers for Each Organization**: Within each organization, you’ll designate certain peers as "anchor peers." **Anchor peers** are specific nodes that serve as primary points of communication between different organizations within the network, helping to ensure the network is well-connected.
3. **Admin User and Peer Management**: By default, the crypto agent (a tool for managing cryptographic materials) generates the cryptographic materials (e.g., certificates) for an "admin" user within each organization. This admin user has the authority to manage the peers within that organization, handling tasks such as adding or removing peers or setting policies.
4. **Defining Additional Users**: Besides the default admin, you have the option to add more users. To do this, you specify an "account" configuration in the configuration file, where you can set up additional users if needed.
5. **Naming of Additional Users**: The "account" value you set determines the number of extra users and their naming convention. For example, if you set "count" to 2, two additional users (User1 and User2) will be created. The crypto agent will then generate separate cryptographic material for each of these users.

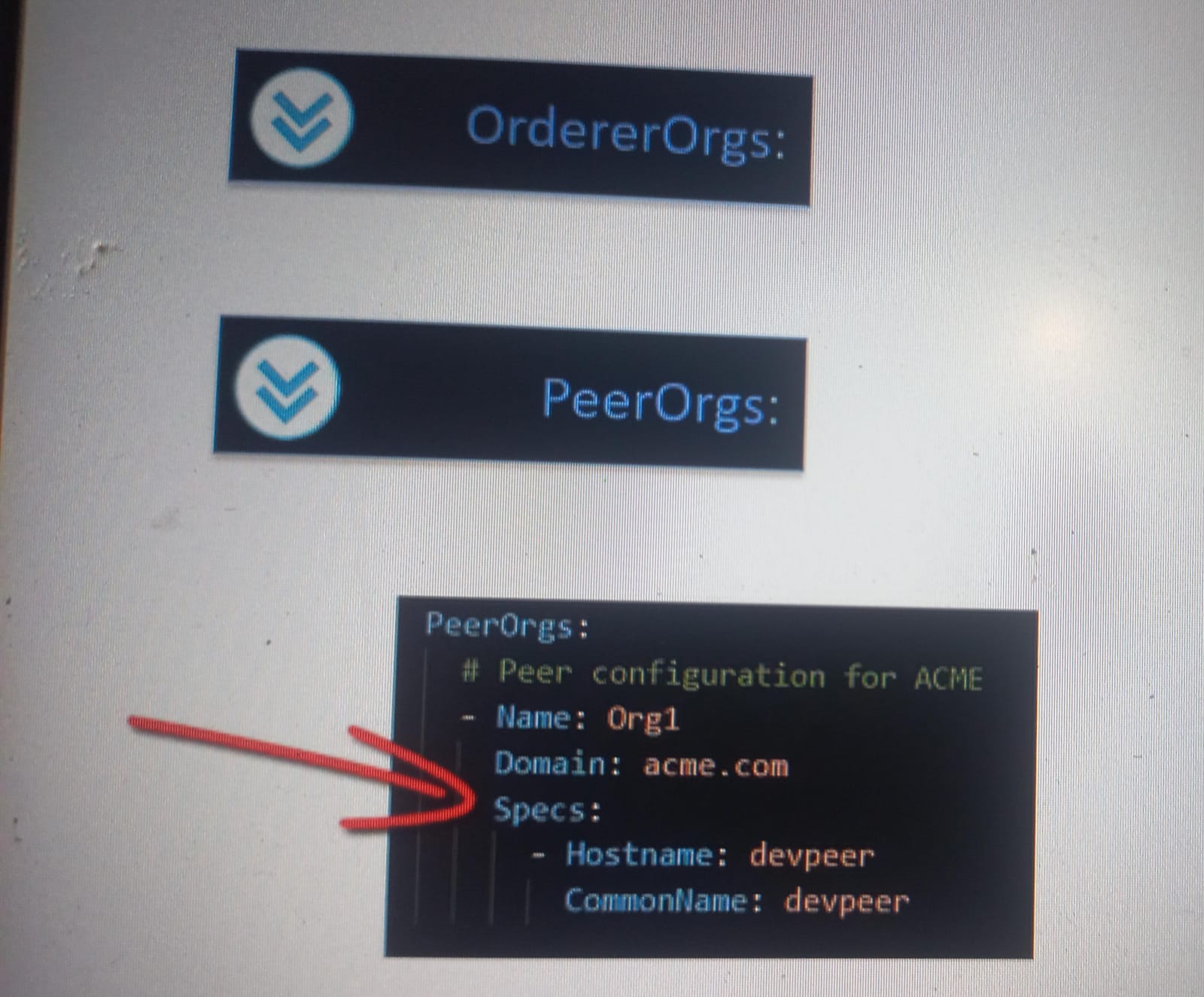
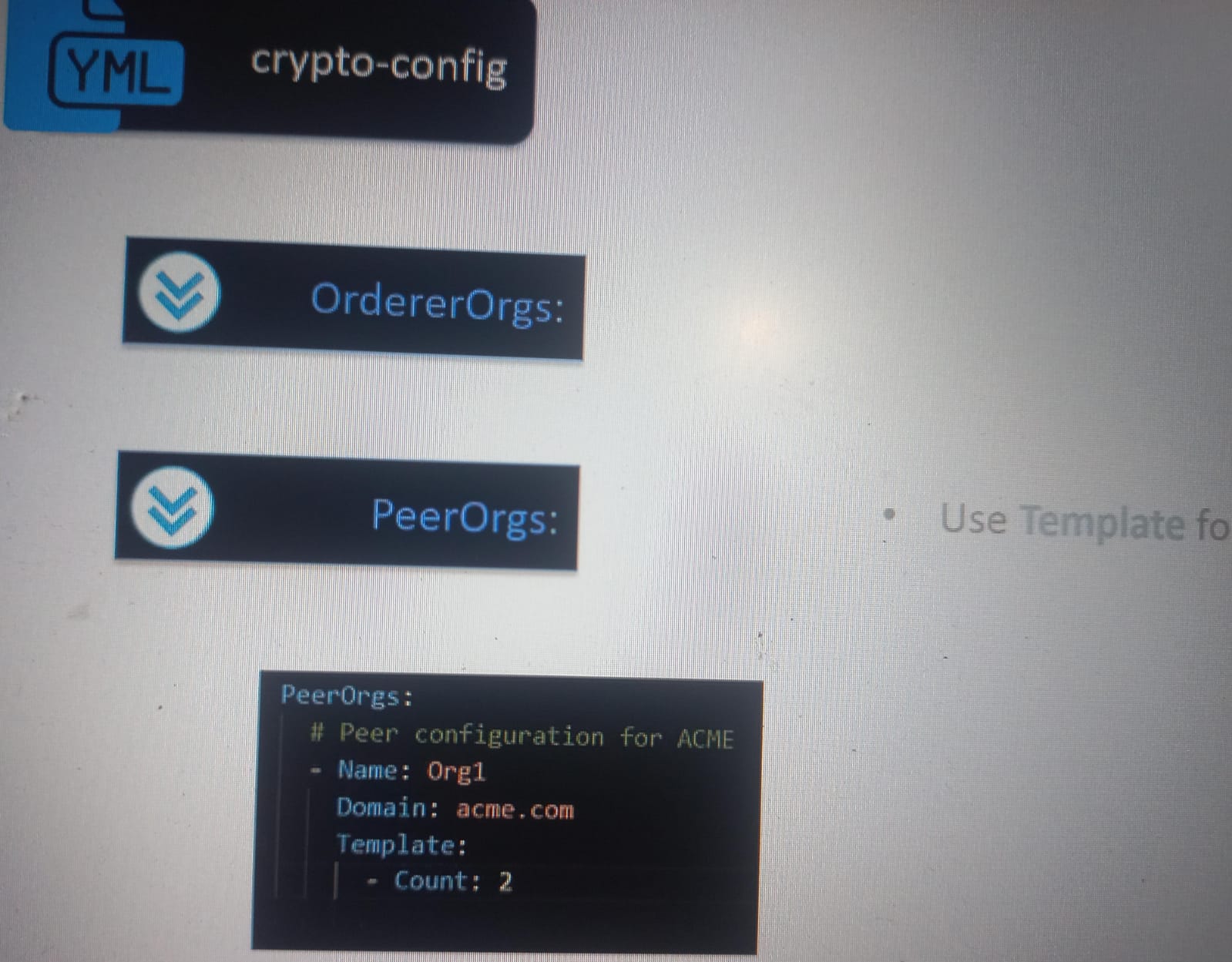


So this is how the peer orgs section will look like for the org one, which I have defined in the sample setup.

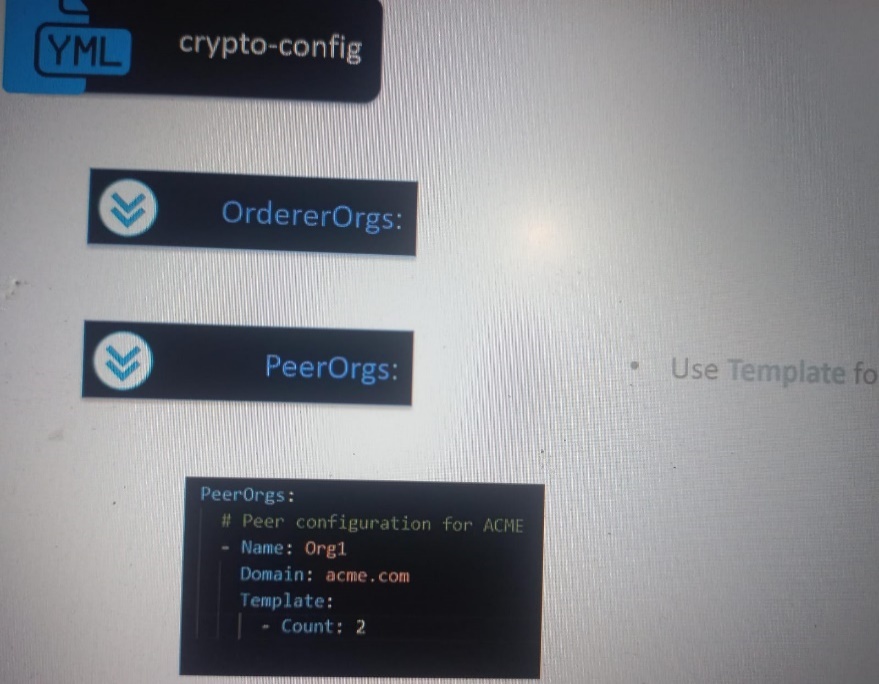
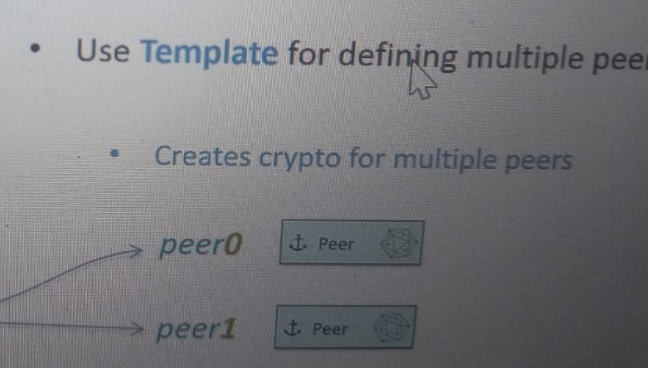
* The name is org one domain is acme.com.
* Now there are two ways to define the anchor peers within the organization.
* specifications way
* the template way.
* You would use the specifications way for defining peers with host names
* So for example, spec section here has a single peer which has a host name dev peer.
* So I can define multiple peers with host names.
* The other way is to go with the template section within the peer definition and that you can use for defining peers without the host names.

Let me explain how that works.

* So for that we will replace the specs subsection with template and account

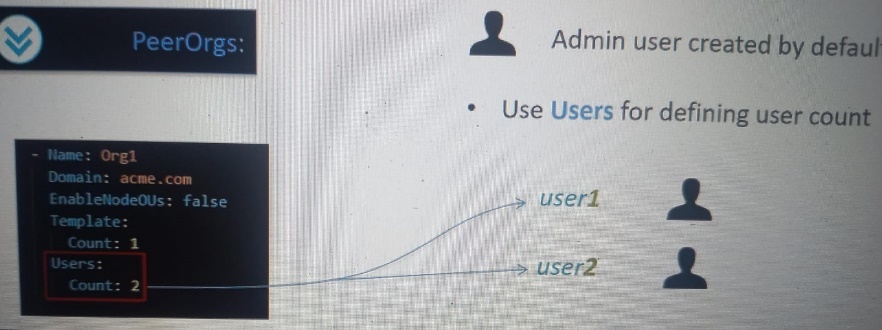
* when we use the template instead of specs the way it works is that count is used for generating the crypto material for multiple peers. How many will depend on the count
* So in this case there will be **two peers** that will get created, peer zero and peer one, and that's the naming convention used for the peer names

* The crypto tool by default generates the crypto material for a special user called the admin user.
* But you can define multiple users as part of the configuration and to do that you would use the users subsection within the organization.

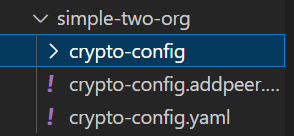
So for example, here the user count is two.

* When you will run this configuration through the crypto tool, it will generate two users user one and user two.

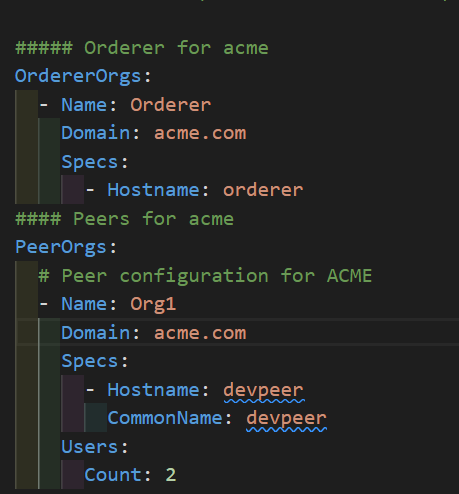


So the naming convention is user followed by a number and the count decides how many users will be created

copy the crypto config.yaml file from crypto.1 sub folder from this location to the simple -two-org sub folder under cryptogen.

So this is how the crypto dash config dot yaml file looks like after I've added the peer orgs section to it here.



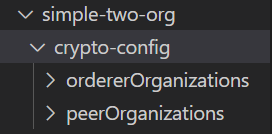
As you can see we have:

* One Org.
* There is one peer, the host name of that peer is dev peer
* the users count is two.

run the **Command->**cryptogen generate --config=./crypto-config.yaml

Now this time you will see there is :

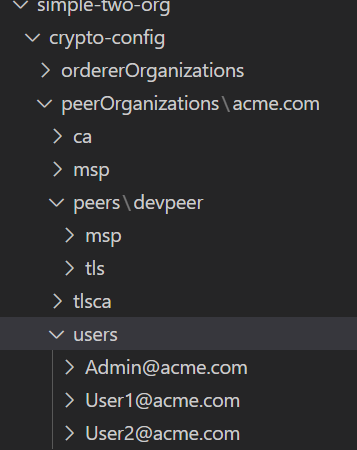
* order organization and
* peer organization



**within the peer organizations:**

you will see that there is one peer dev, peer and within the users you will see the special user,the admin user, user one and user two.

There are two users as defined by the count.



**Summary**

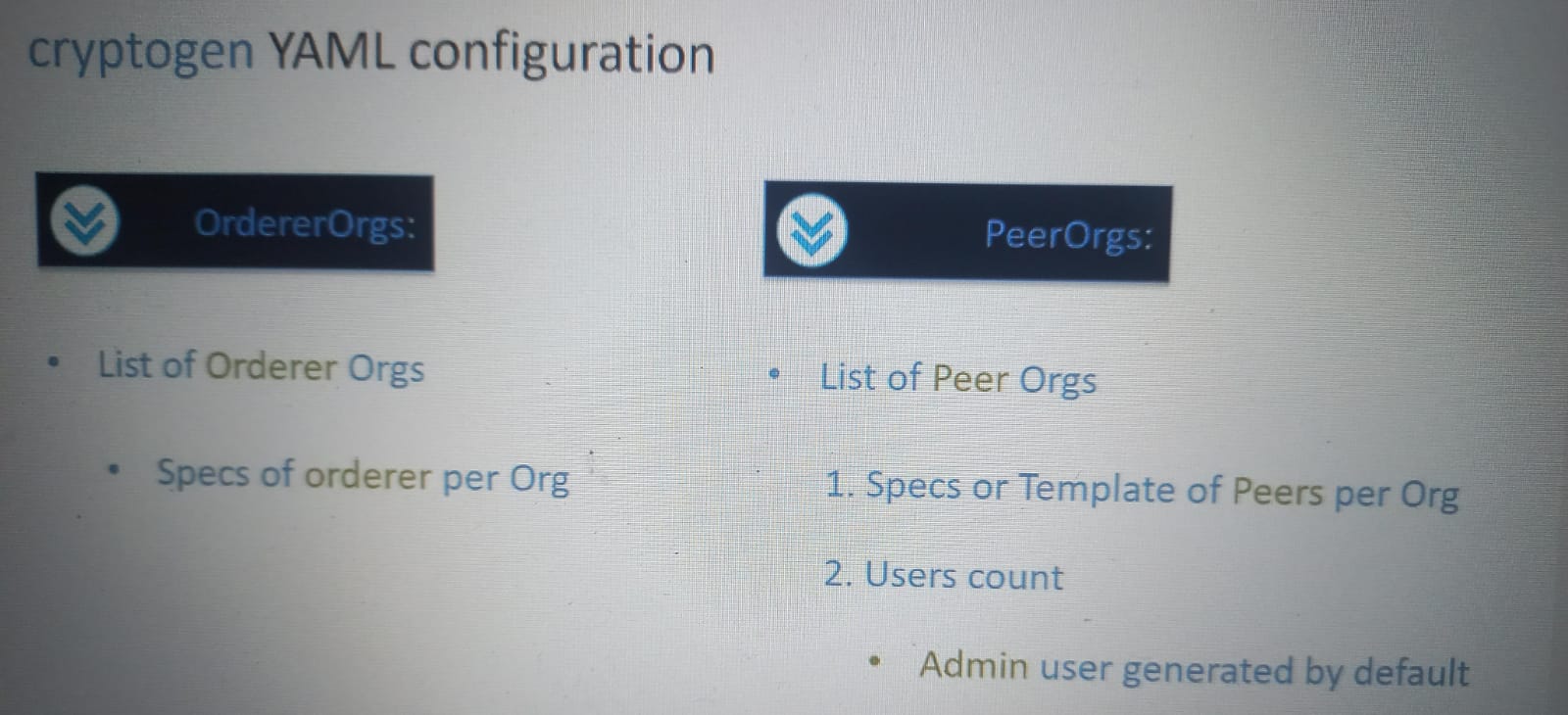
* I talked about how to create Yaml configuration for the crypto gen tool.
* There are **two sections** in the Yaml configuration for crypto gen

1. the **order orgs** in which:

* you will declare or define the organizations that are hosting the order.
* order specification have to be provided on per organization basis.

1. **peer orgs** section in which:

* you will define the set of organizations which are hosting the anchor peers
* Specifications or the template of peers need to be provided on per organization basis so there can be one or more peers per organization.
* You can also define the count of users for which you would like to generate the crypto material.
* By default, the crypto gen tool generates an admin user, so the user count is in addition to that one admin user.



**Test environment setup & use of cypto material**

Learn about:

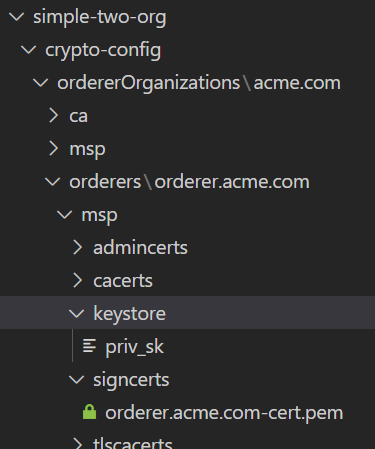
1. the various options available for setting up test environment
2. how this test environment uses the test crypto material.
3. how to extend the configuration.

* For example, if you want to add a new peer to your test configuration, how can you do that without overwriting an existing configuration?

Before I start with the discussion on how the crypto material is used by the orderer and the peers,

**let's take a quick look at the crypto material generated by Cryptogen.**

Here I've expanded the folder for the Orderer organizations.



Admin’s Public Key

Private Key

Public Key/Certificate

Transport Layer Security-Certs/Keystore

* Under the order subdirectory there is a **order.acme.com** that has the crypto material for the order in the configuration that we have created in the previous Heading.
* So here under the **Msp folder** you will find there is a **sign certs sub folder** which has the **Pem file** that is **the public key or the certificate for the order.**
* The **Keystore sub folder** has the private key for the order.
* This private and the public key are used by the order binary for communicating with the other components within the Hyperledger fabric infrastructure.
* The **admin certs sub folder** has the admins Public key
* admin is the one who would take the administrative action on the orders organization and their certificate. The public certificate is available under this sub folder.
* If the transport layer security is enabled, then you need to use the TLS certificates and key stores and those are available under the TLS sub folder.

At this point, I would suggest that you browse through the crypto material on your own just to get a feel of the generated components.

**Orderer needs access to the MSP and TLS:**

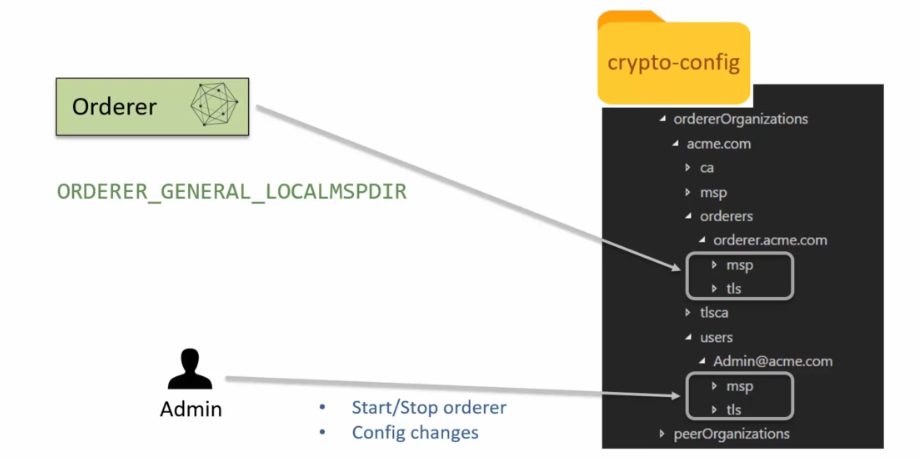
The order binary setup requires access to the certificates and the keystore for the order which is available under the **Msp** directory.

If the transport layer security or the TLS is enabled, then the order binary will also require access to the TLS certificate and the Keystore.

This access is enabled by way of using the environment variables or by way of the configuration file.

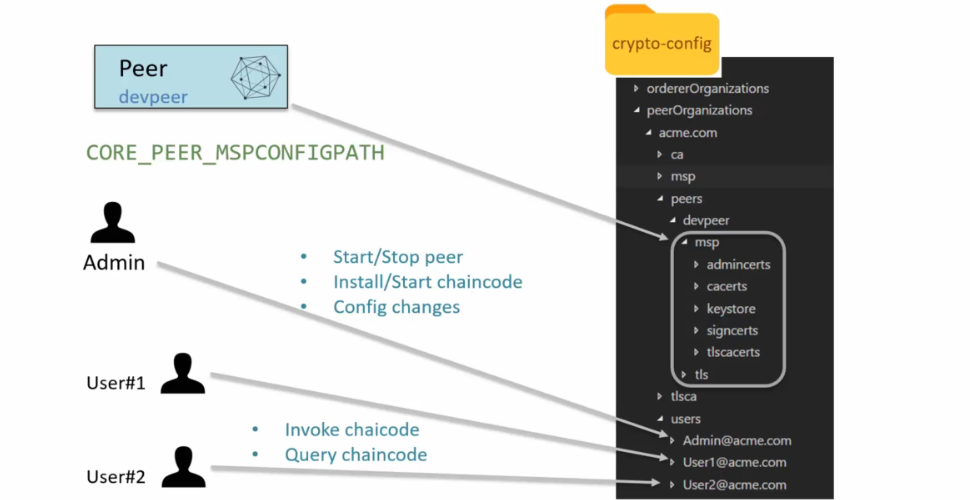
just assume that the order binary needs to access the certificates in the **MSP and the TLS directory.**

The administrator of the organization, which is created by default, also has the MSP and the TLS directory and to execute the command by way of the order and the peer binaries, the administrator need to setup access to the certificate within these two directories.



**Peer needs access to the MSP:**

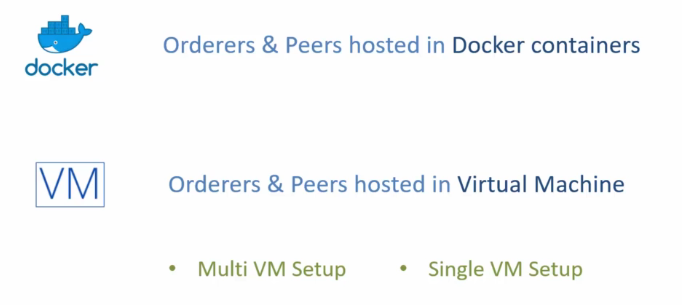
* like the order setup The peer binary setup also requires access to the certificates available in the MSP and the TLS directory,
* so Peer will be configured to access the certificates and the Keystore under MSP.
* Same way the administrator will be using the certificates and keystore available under the admin directory.
* They will use that certificate to use the peer binary, for example, for starting or stopping the peer for installing and starting the chain code and for making configuration changes.
* The users will use the certificates available under the user 1 and user 2 directory.
* The users will use these certificates for invoking the chain code and for querying the chain code.



There is no hard and fast rule on how you need to set up your Hyperledger fabric test environment.

You can set it up using **Docker containers** wherein the orderers and peers in the infrastructure are hosted in Docker container instances.

Or you may use **virtual machines** for hosting these different components.

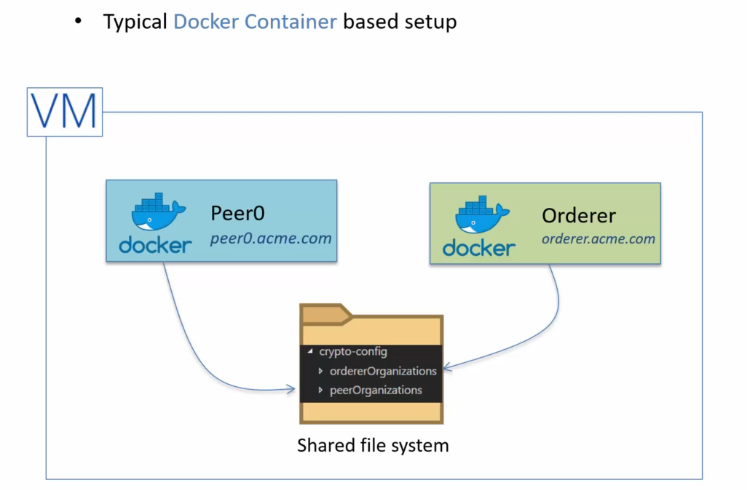


* Now, when you're using a virtual machine, you can have multiple virtual machines set up in which each of these orders and peers are hosted in their own virtual machine.
* Or you may use a single virtual machine to install all the peers and orders.
* Obviously there are certain aspects that you will have to deal with when you are using a single VM setup.

**how our Docker container based test environment can use the test cryptomaterial:**

**DOCKER CONTAINER BASED SETUP:**

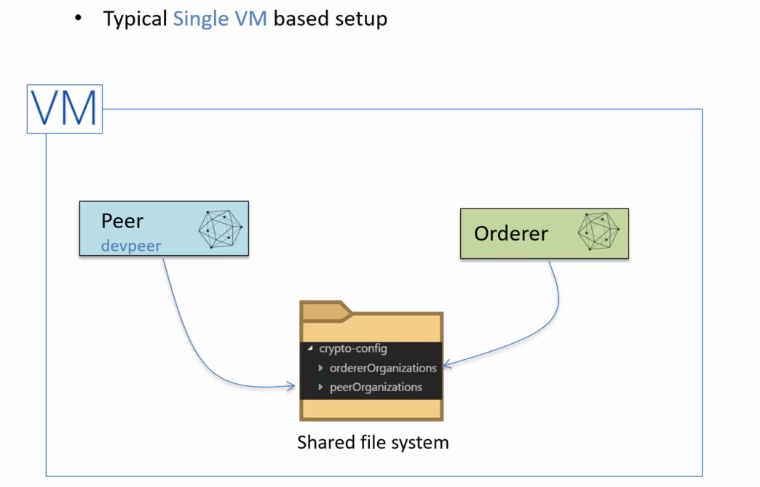
* Typically the way it will work is you would provision a virtual machine, install Docker on it.
* At the runtime, there will be a container for the peer,
* and this container for the peer will have access to the peer organizations, MSP crypto material that is the certificates and the Keystore.
* Similarly, there will be a container for the order that will have access to the MSP subfolder in the order organizations.
* The crypto config folder will be on the VM and will be shared between the peer and the order.



* In a typical multi virtual machine setup, the peer and the order will reside in their own instances of the virtual machine and
* to make the MSP crypto material available to these Components You will have to copy the Msp for the order to the order virtual machine and
* then you'll have to copy the peer Msp to the peer virtual Machine.
* Now you may also use a shared folder between the virtual machine, but this mimics a real production like environment.

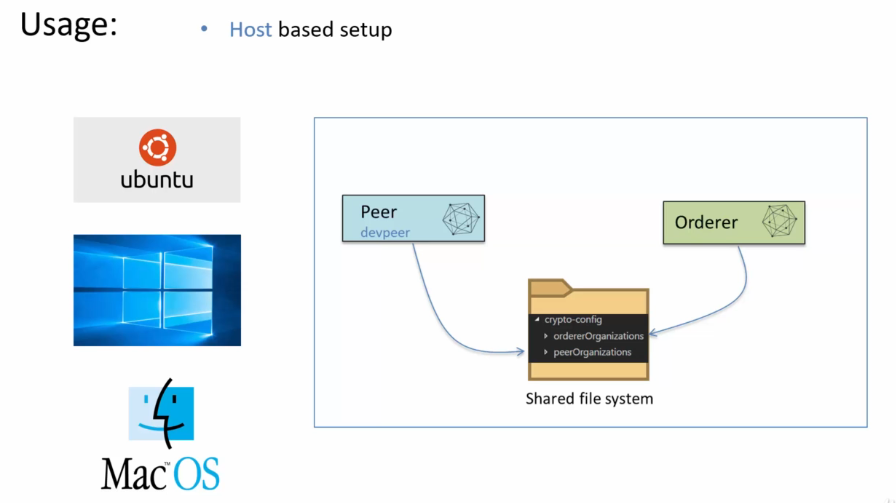
**SINGLE VM BASED SETUP:**

* In a typical single virtual machine based test setup, you would provision a virtual machine and
* Then launch the native pier binary for creating a peer process.
* Now, this peer process will need access to the Msp crypto material which will be available under the crypto config folder on the file system.
* Similarly, the order process will be launched by using the order binary and the order will need access to the order organizations Msp, crypto material.
* And so in this case we're talking about a single machine with natively hosted processes.
* The crypto config folder, which is on the native file system, is shared by the peer and the order process.
* So you will see that this is very similar to the Docker setup, **except that the containers are not in use.**

****

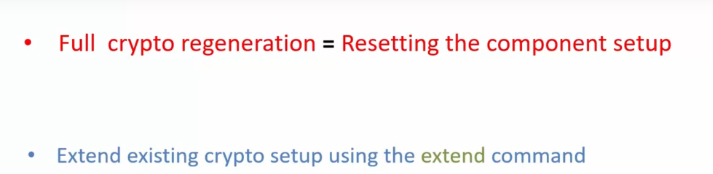
**HOST BASED SETUP:**

* Another common setup is the host based setup in which you install the binaries on your host machine, which can be Ubuntu windows or Mac OS.
* Once the binaries are installed, you will see that the setup looks exactly like a single VM setup.
* So this is another way of creating a test environment on your own machine.

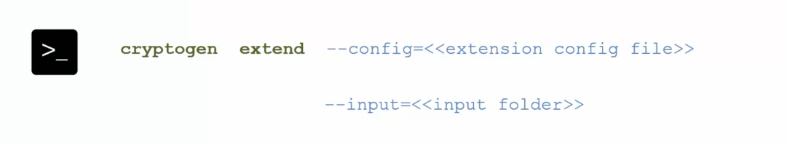


**ADDITION OF COMPONENTS TO EXISTING SETUP:**

* Sometime as part of your testing or experimentation, you may need to extend your setup to include new organizations or peers or other components.
* In such scenario, you will have to change the crypto config file and when you would do that and run the cryptogen tool on it, it will regenerate all the crypto material.
* As a result, it will require you to reset your components within the test environment
* to avoid such a situation What you can do is you can use the **extend command** of the crypto gen tool.



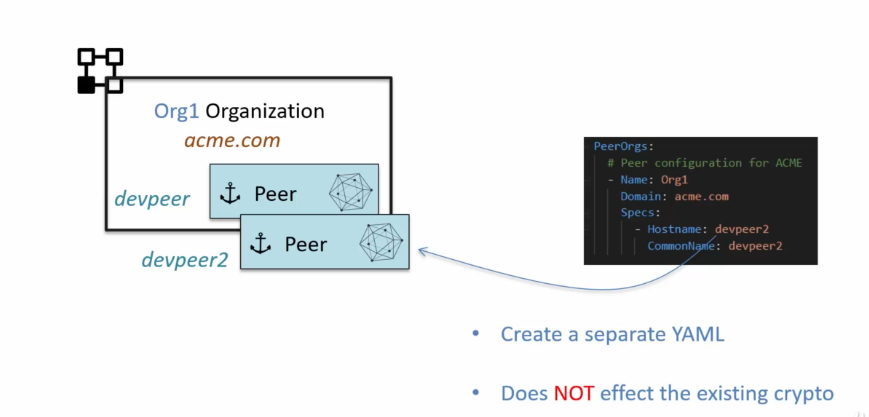
* The Extend command works like this.
* You would call cryptogen extend -- config and then you will provide the template file with the additional component.
* So this is the extension configuration file, not your original configuration file.
* And then you would provide the existing crypto gen generated folder.
* By default, this is the crypto-config folder.
* So if you do not provide --input, the crypto gen tool will expect the crypto -config folder to be available.



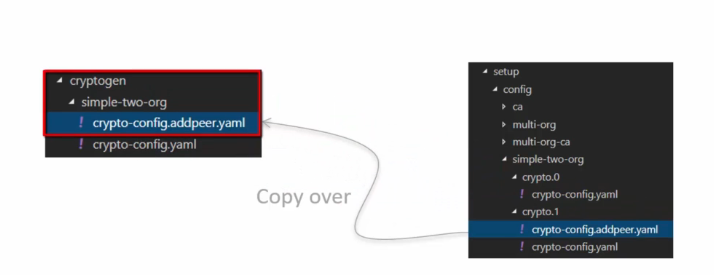
* when this command has executed successfully, the appropriate crypto material will appear under the sub folders within the crypto dash config folder.
* You can use this command to
* Addition of a member(organization)
* adding orderers,
* adding peers to an organization
* for adding users.

**ADD A PEER FOR ORGANIZATION**

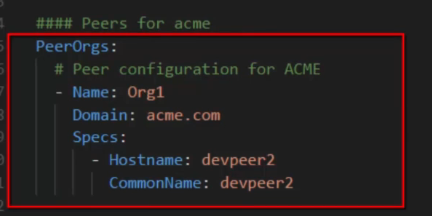
* We will add a peer for an organization to the existing crypto setup.
* So we already have the org 1 organization in which there is a peer by the name Dev Peer.
* we're going to create a new dev peer, we'll call it dev Peer2
* when we will run the cryptogen command on the extension configuration, it will generate the crypto material for dev peer 2 under the crypto -config folder.
* The key points to note here is that this extension file is not the same as the original crypto config file that we used This is a new file
* once you run the extend command it will not affect the existing crypto material.



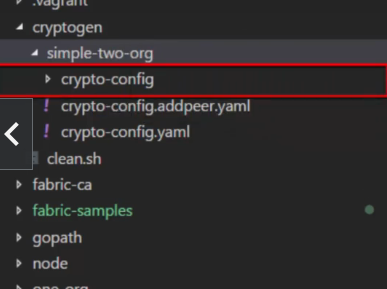
The crypto -config.add peer.Yaml file is available here and it needs to be copied to the crypto gen Simple dash two org sub folder.



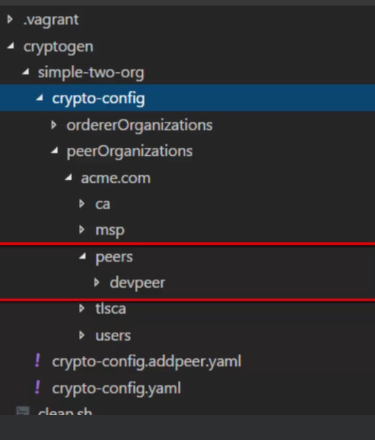
1. In this as you can see, I have peer orgs name of the organization is org1.Domain is Acme.com. And from the specifications perspective, we have a host name and the common name set to dev peer two.



1. Now we already have the crypto config folder that was generated as part of the cryptogen generate command.



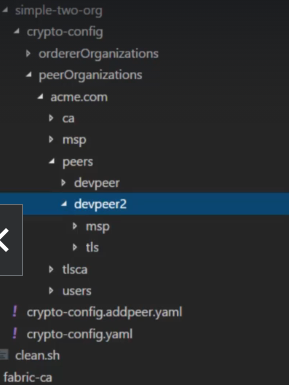
1. And as you can see under the peers sub folder, we only have the dev peer.



1. So what we're going to do next is run the extend command to generate the crypto material for dev peer to crypto gen extend --config and this will be set to the full path to the crypto config.add peer. yaml.

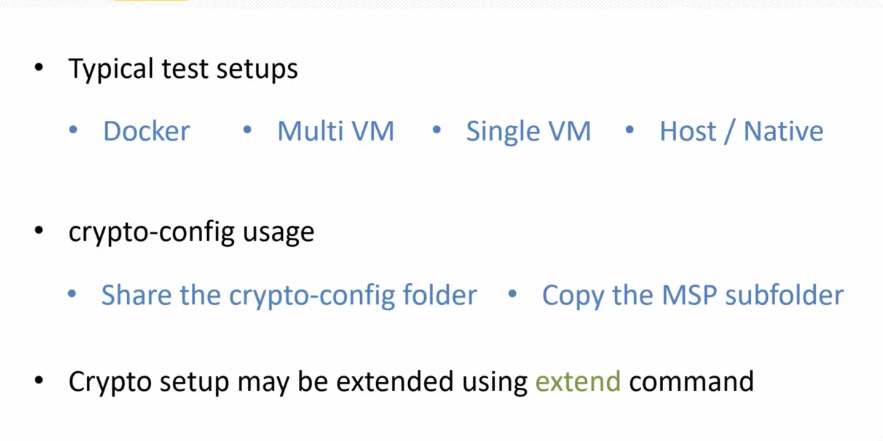
**Command to Run:** **cryptogen extend --config=./crypto-config.addpeer.yaml –input=./crypto-config**

1. Now the input folder is although we don't need to give it, but you can use the --input crypto config by default it is crypto config.
2. So after the successful execution we have the crypto material available for the dev 2 peer under the crypto -config folder.



**Summary**

* Talked about the typical test setups.
* You can set up your test environments by
* Docker containers.
* multi virtual machine
* the single virtual machine or
* similar to the single virtual machine (Host/Native) You can install the binaries for Hyperledger fabric on your own Windows Ubuntu or Mac OS machine to set up your test environment.
* Each of these setups require access to the crypto material.
* You can either share the crypto config folder with the various components in the test server in the test setup,
* or you can copy the msp sub folder for the component to the VM or to even the container.
* Often you have to extend the test setup and to do that you can use the extend command available for crypto gen.   
  So this way you will not overwrite the existing crypto gen generated crypto material.



**Exercise - Setup crypto-config.yaml for 2 Peer Orgs**

**1 Peer:**

# Defines two organizations

#  - Orderer organization

#  - Org1  = Single peer organization

#    Uses the "Specs" to declare the peers instead of Template

##### Orderer for acme

OrdererOrgs:

  - Name: Orderer

    Domain: acme.com

    Specs:

      - Hostname: orderer

#### Peers for acme

PeerOrgs:

  # Peer configuration for ACME

  - Name: Org1

    Domain: acme.com

    Specs:

      - Hostname: devpeer

        CommonName: devpeer

    Users:

      Count: 2

**2 Peer:**

# Defines three organizations

#  - Orderer organization

#  - Org1  = Single peer organization

#    Uses the "Specs" to declare the peers instead of Template

#  - Org2  = Another peer organization

# SOLUTION to the exercise

##### Orderer for acme

OrdererOrgs:

  - Name: Orderer

    Domain: acme.com

    #EnableNodeOUs: true

    Specs:

      - Hostname: orderer

#### Peers for acme

PeerOrgs:

  # Peer configuration for ACME

  - Name: Org1

    Domain: acme.com

    #EnableNodeOUs: true

    Specs:

      - Hostname: devpeer

        CommonName: devpeer

    Users:

      Count: 2

  # Peer configuration for Budget

  - Name: Org2

    Domain: budget.com

    #EnableNodeOUs: true

    Specs:

      - Hostname: devpeer2

        CommonName: devpeer2

    Users:

      Count: 1