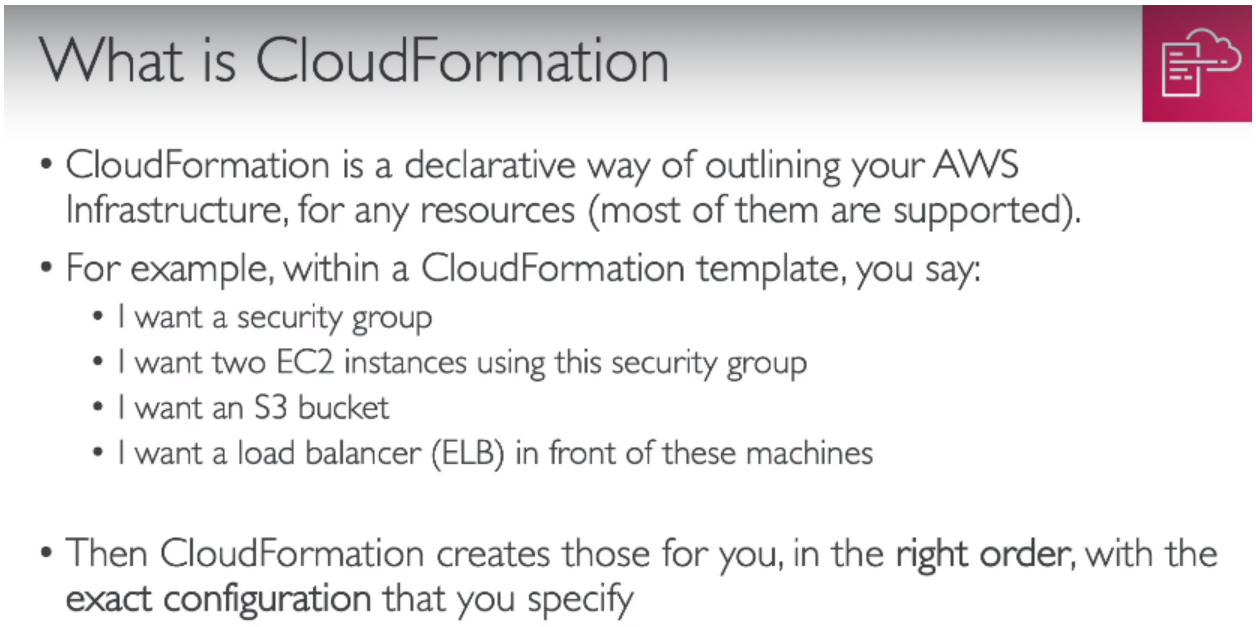
CloudFormation Overview



1.

2. Sure. AWS CloudFormation is an Infrastructure as Code (IaC) service that allows you to create and manage a collection of related AWS resources. You can use CloudFormation to model your infrastructure as a collection of resources, and then use CloudFormation to provision and update those resources as a single unit.

   For example, you could use CloudFormation to create a stack that includes an Amazon EC2 instance, an Amazon S3 bucket, and an Amazon Relational Database Service (RDS) database. You could then use CloudFormation to update the stack to add a new EC2 instance or to change the size of the RDS database.

   CloudFormation templates are written in JSON or YAML format, and they describe the resources that you want to create and the properties of those resources. When you create a stack, CloudFormation uses the template to create the resources and then configures them according to the properties that you specified.

Benefits of AWS CloudFormation (1/2)

- Infrastructure as code
  - No resources are manually created, which is excellent for control
  - Changes to the infrastructure are reviewed through code
- Cost
  - Each resources within the stack is tagged with an identifier so you can easily see how much a stack costs you
  - You can estimate the costs of your resources using the CloudFormation template
  - Savings strategy: In Dev, you could automation deletion of templates at 5 PM and recreated at 8 AM, safely

3.
4. Saving Strategy..We can close our cloud formation template in particular time and again create it when needed
5. The cloud formation knows what to create in the sequence …

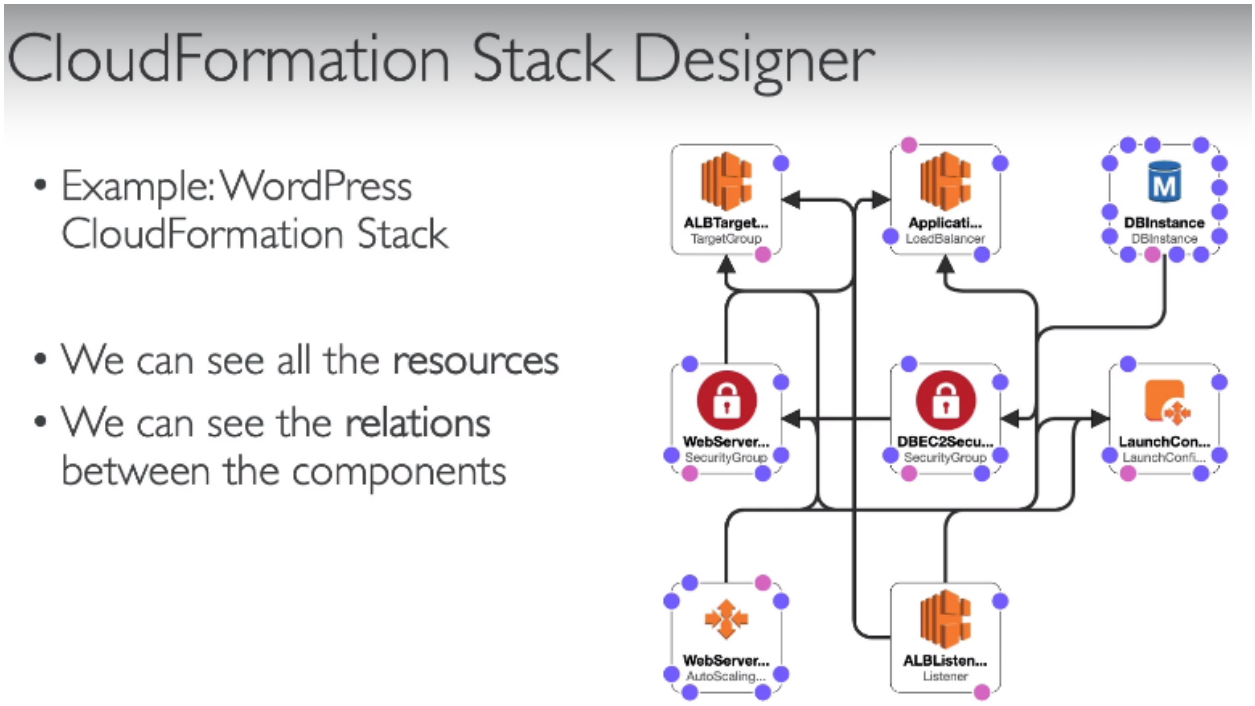
Benefits of AWS CloudFormation (2/2)

- Productivity
  - Ability to destroy and re-create an infrastructure on the cloud on the fly
  - Automated generation of Diagram for your templates!
  - Declarative programming (no need to figure out ordering and orchestration)
- Don't re-invent the wheel
  - Leverage existing templates on the web!
  - Leverage the documentation
- Supports (almost) all AWS resources:

6.

- Supports (almost) all AWS resources:
  - Everything we'll see in this course is supported
  - You can use "custom resources" for resources that are not supported

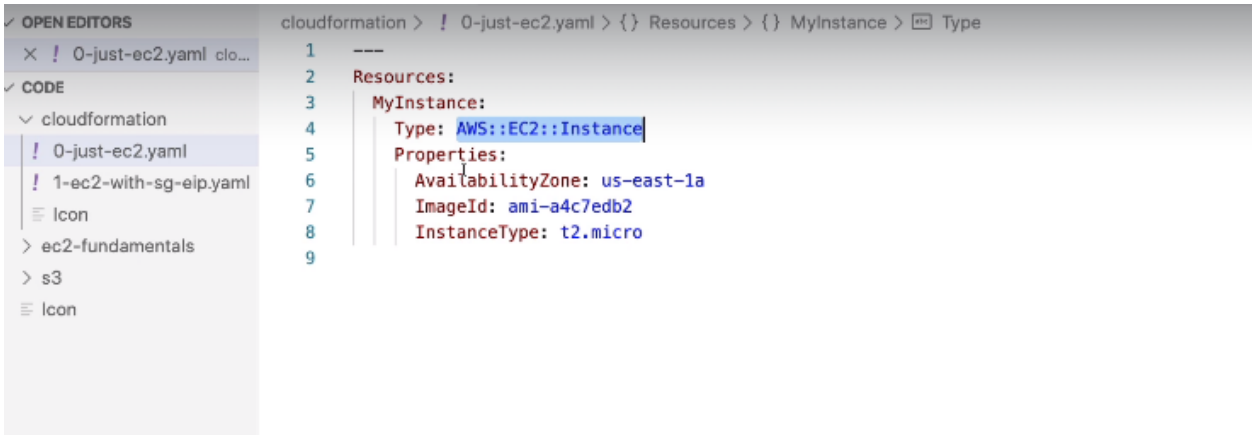
7.



8.
9. Here in the diagram ..cloudformation created 2 security grps,ALB Target grps, Auto scaling..etc…see the diagram

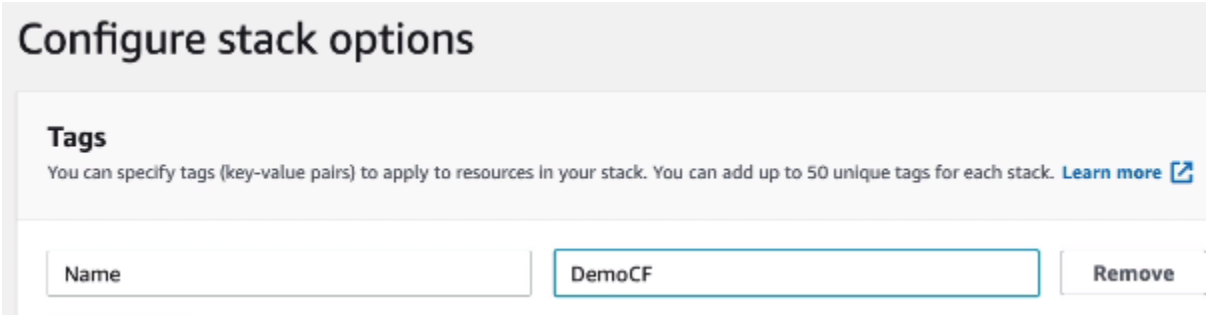
Cloud Formation Hands ON:

1. First we go to aws cloud formation service…and click on create template
2. Here we have a template and we choose template is ready option
3. And we will upload our cloudformation template…0-just-ec2
4. It is a basic template..which creates an ec2 instance
5. Here imageid is AMI id

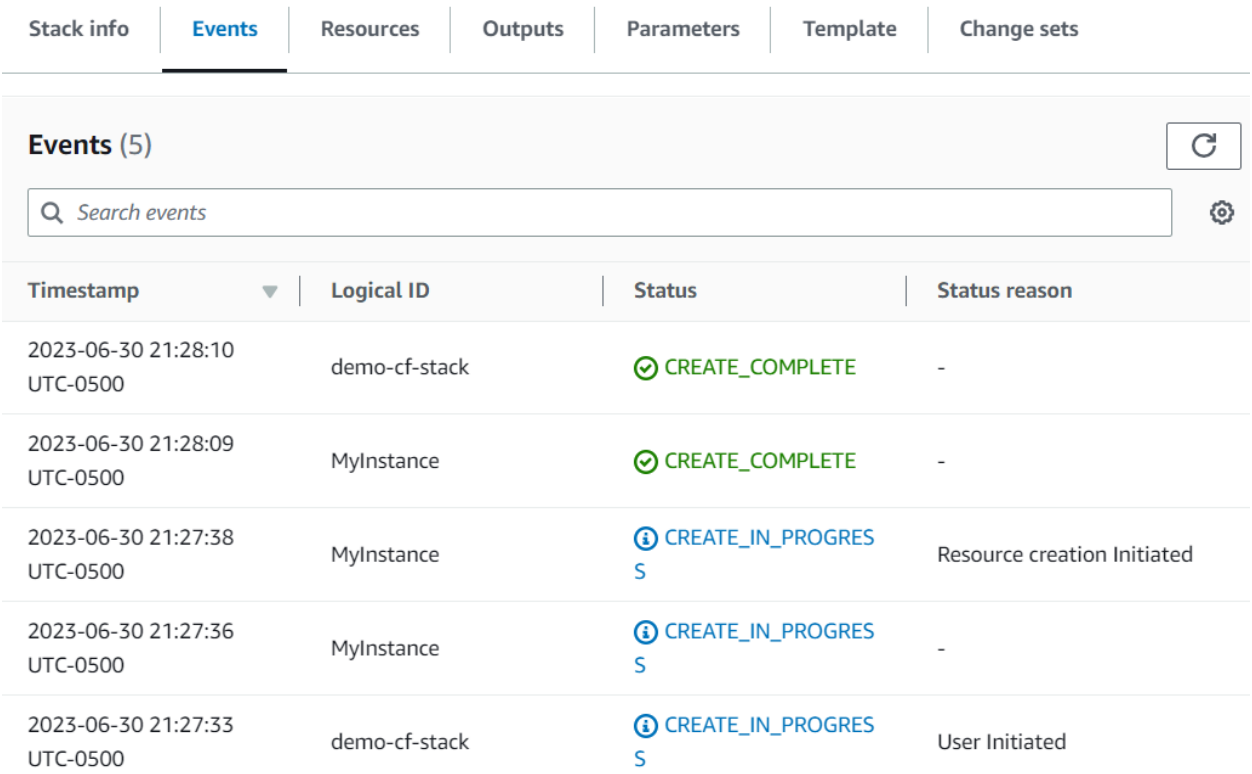


6.
7. We can also view our template in designer mode

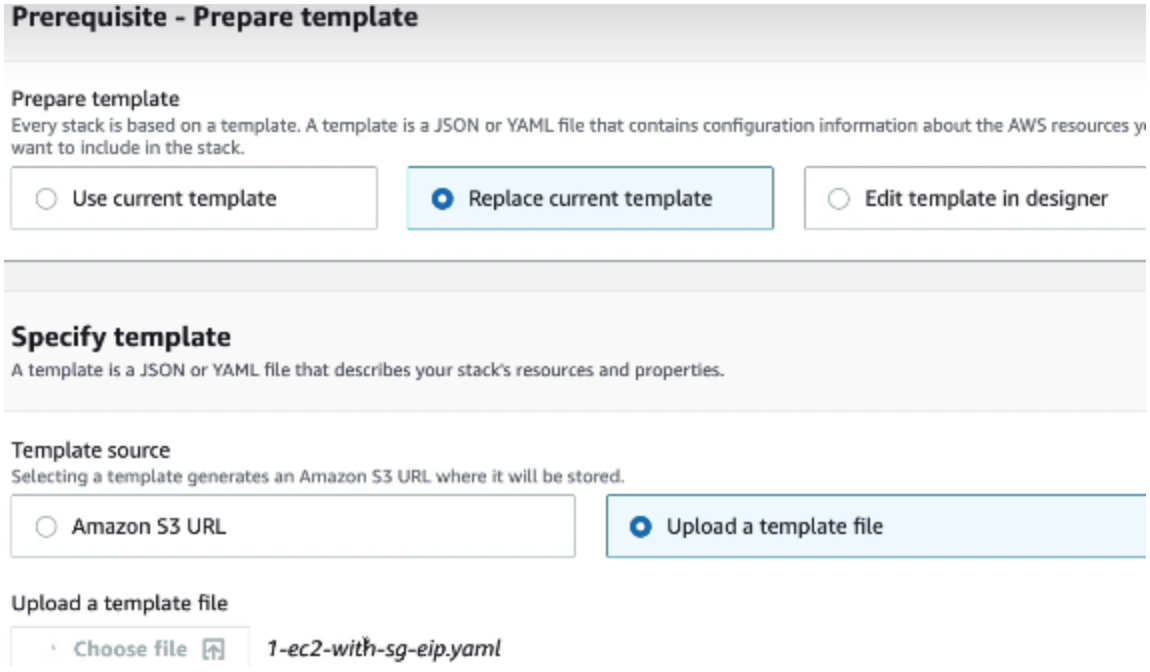
8. Click on next..in this page we can add tags to our temple



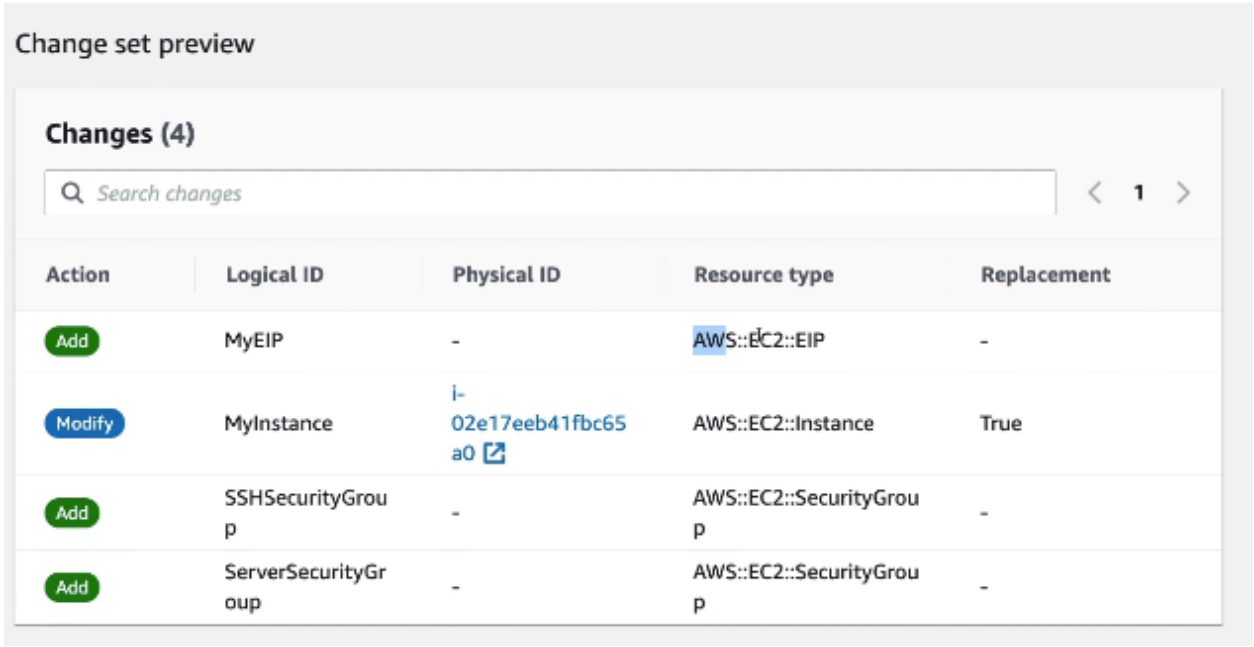
9. In the next step..we can review everything and click on create stack
10. And as you can see what I did is that I uploaded a template so the template is here, this is just some lines of codes. And this template is actually going to be used to create an EC2 Instance. So this is why CloudFormation is called Infrastructure as Code.
11. We can also check events



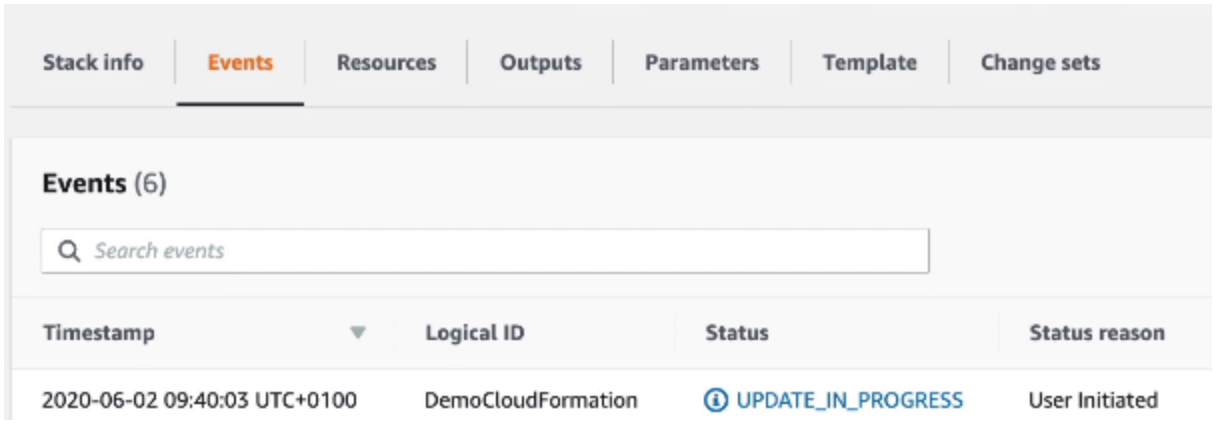
12. After the creation of instance..our stack completed its work..
13. Now we work in our ec2 instance
14. We can also update our current template

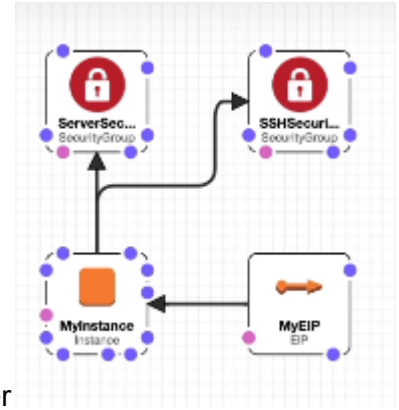


15.
16. Click in update template and go to replace current template and upload an existing template from local disk..here we uploaded 1-ec2-with-sg..
17. If we go through template code..here we have security grps and all ..
18. Next we will create a new tag for it and in the last we have change set preview
19. It say what it is going to change in our cloud formation from previous template



20.
21. In here it is modifying template and adding security groups and EIP
22. Next click on update stack
23. Now in events we can that our cloudformation is updating

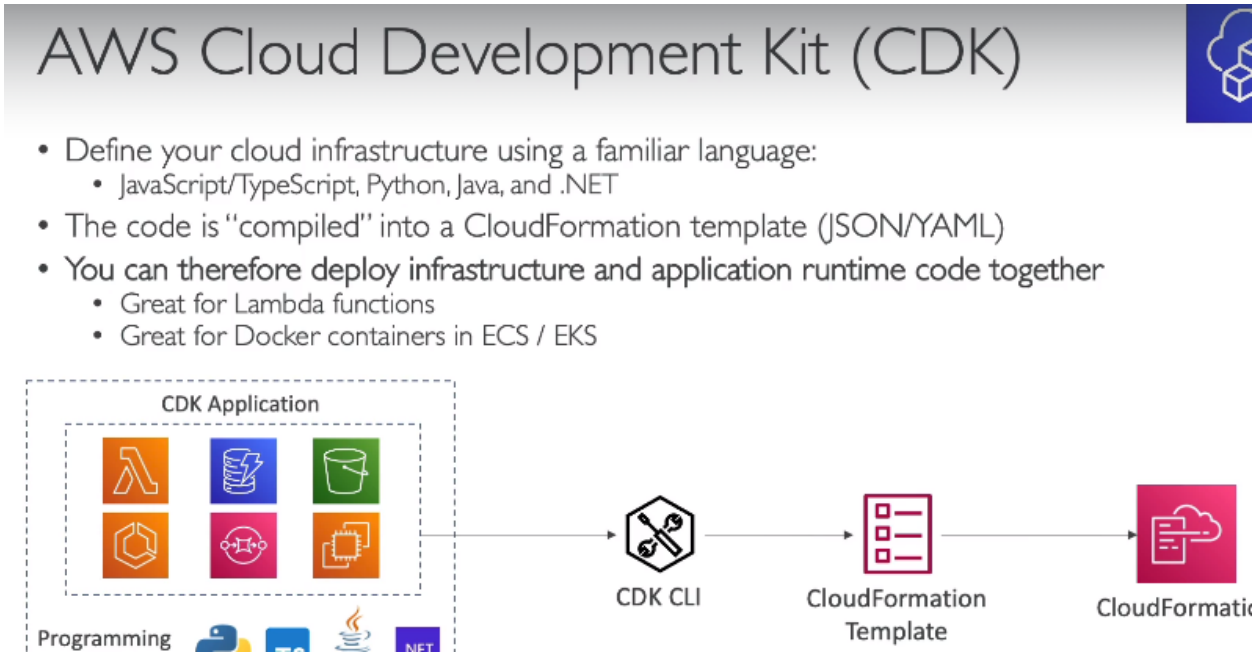
24. 



25. We can also view our template in designer
26. In resources section ..we can see what resources are allocated for our template
27. If the current updating is finished…the previous instances will be deleted
28. In the end ..if we delete our stack..then entire resources will get deleted slowly…
29. So, all in all, to summarize
30. CloudFormation is a really easy way to define templates and resources and Infrastructure as Code.
31. CloudFormation allows you to take the same template and if it's well written you can deploy it to many AWS regions or many AWS accounts. And that makes it really is a base foundation for so many other types of infrastructure on AWS.

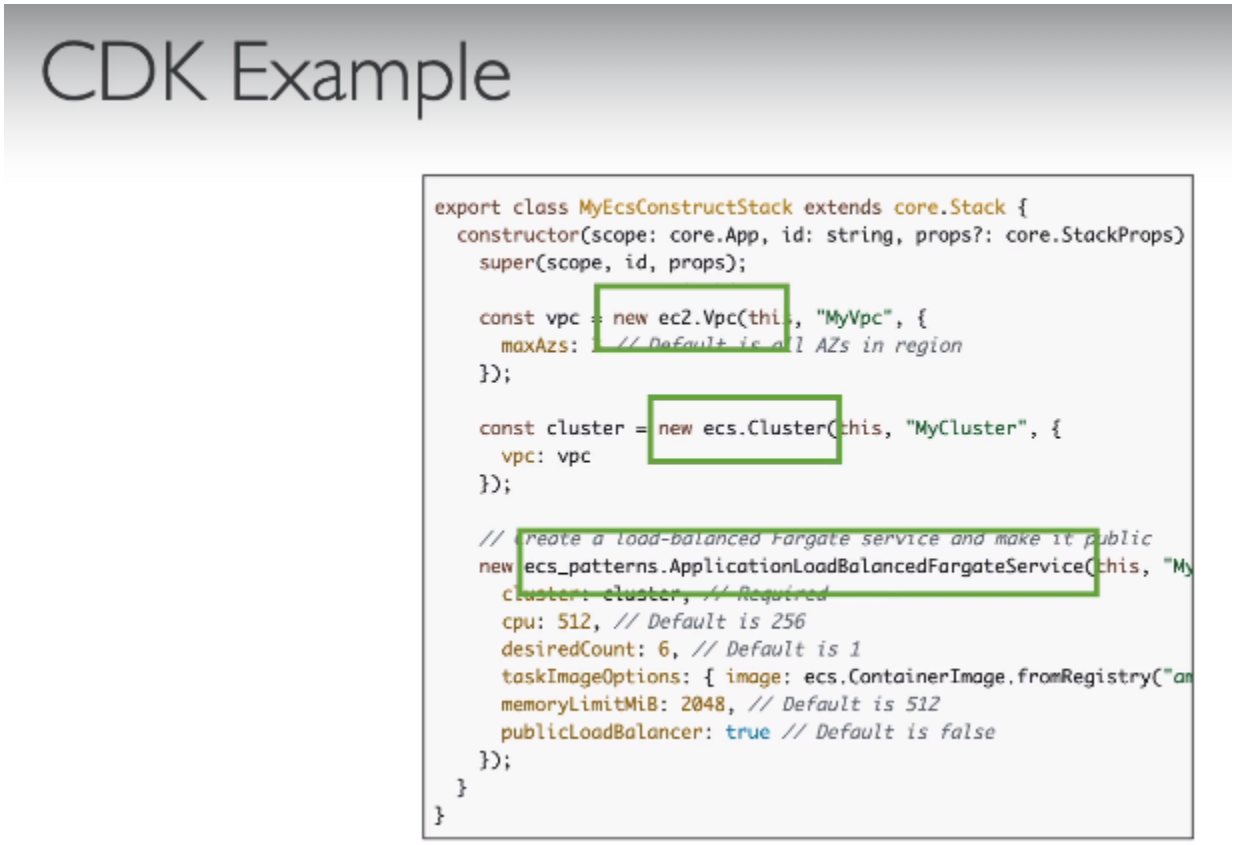
AWS CDK Overview

1. Here we can define our cloud infrastructure with familiar languages..like js,python etc
2. The writtern code..will then compile into cloud formation template(JSON/YAML)

3. 

4. Lets take an example…we choose python
5. So we're going to write our own CDK application in Python, and we're going to do define our Lambda function of that MDB table and maybe for other services in AWS.
6. Now this cdk application..using cdk cli…will transform our python code..to cloudformation template..which is used to generate cloudformations

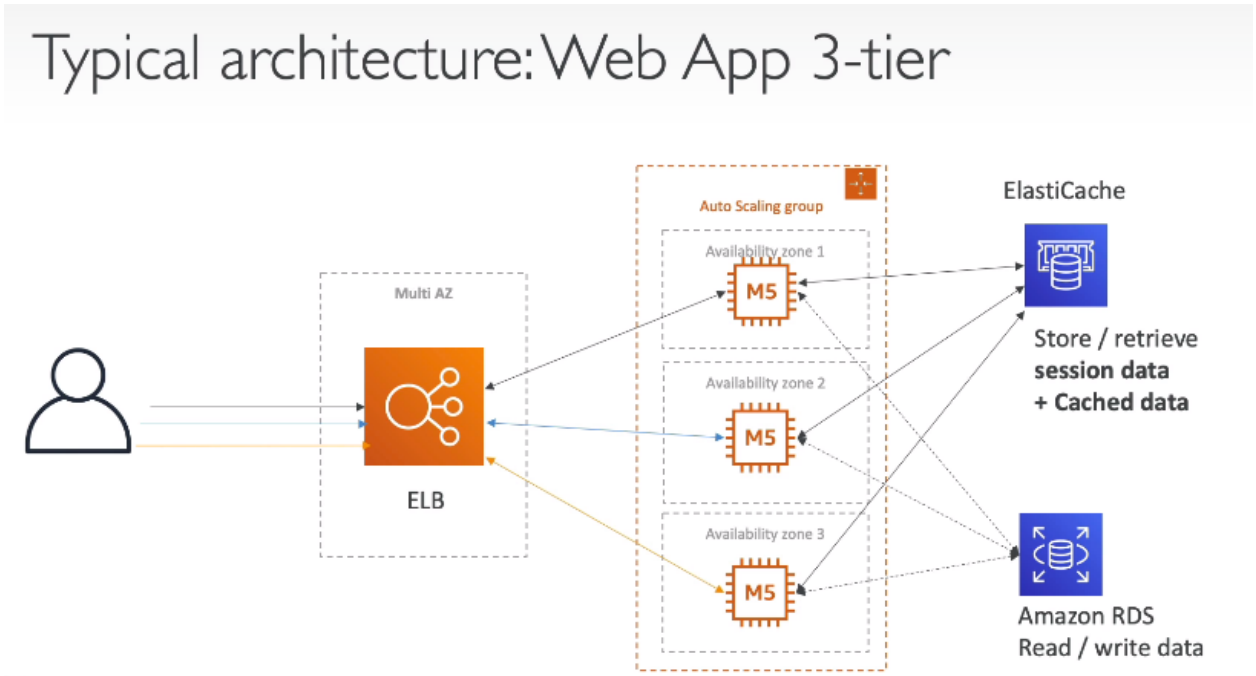
7. A sample cdk example using typescript/js



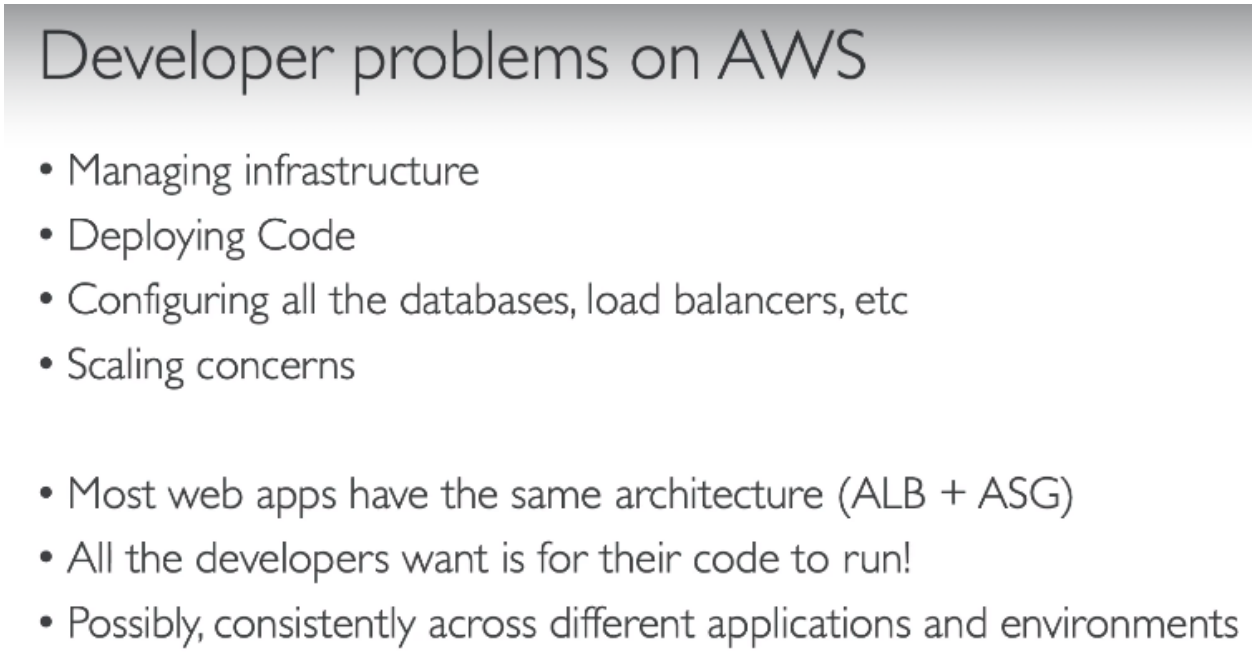
8. Here we have vpc,ecs cluster and application load balancer

Beanstalk Overview:

1. So when we have deployed a web application in AWS, we typically follow a architecture called a three-tier architecture.
2. Here user's send request to the ELB and ELB forwards traffic to different instances which are managed by auto scaling grp and also which are in diff zones..
3. And then these EC2 instances need to store data somewhere,
4. so they will use a database such as Amazon RDS for relational database to read and write data
5. if they need to have an in-memory database or an in-memory cache, then they can also use elastic cache to store and retrieve the session data or the cached data.



6.
7. We can reproduce this architecture manually..but there is better way in aws
8. So when you're a developer on AWS, you don't want to be managing infrastructure.
9. You just want to be deploying code,
10. You don't wanna be able to configure all databases, load balancers, et cetera.
11. As a dev ..we need to focus mainly on code



12.
13. Here comes aws elastic beanstalk

AWS Elastic Beanstalk Overview

- Elastic Beanstalk is a developer centric view of deploying an application on AWS
- It uses all the component's we've seen before: EC2, ASG, ELB, RDS, etc…
- But it's all in one view that's easy to make sense of!
- We still have full control over the configuration

- Beanstalk = Platform as a Service (PaaS)

14.

- Beanstalk is free but you pay for the underlying instances

15.

Elastic Beanstalk

- Managed service
  - Instance configuration / OS is handled by Beanstalk
  - Deployment strategy is configurable but performed by Elastic Beanstalk
  - Capacity provisioning
  - Load balancing & auto-scaling
  - Application health-monitoring & responsiveness

- Just the application code is the responsibility of the developer

- Three architecture models:
  - Single Instance deployment: good for dev
  - LB + ASG: great for production or pre-production web applications
  - ASG only: great for non-web apps in production (workers, etc..)

16.

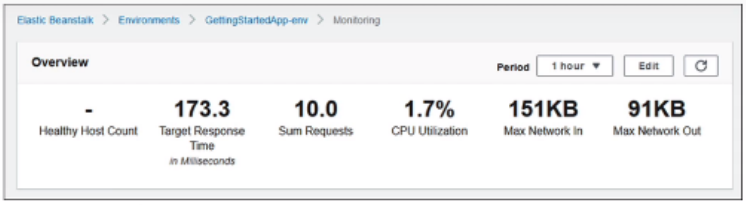
Elastic Beanstalk

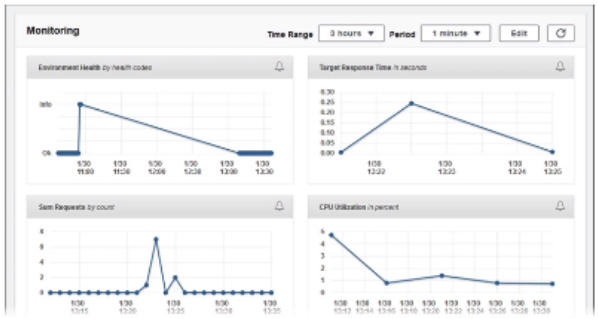
- Support for many platforms:
  - Go
  - Java SE
  - Java with Tomcat
  - .NET on Windows Server with IIS
  - Node.js
  - PHP
  - Python
  - Ruby
  - Packer Builder
  - Single Container Docker
  - Multi-Container Docker
  - Preconfigured Docker
- If not supported, you can write your custom platform (advanced)

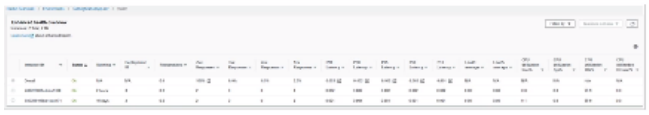
17.

Elastic Beanstalk – Health Monitoring

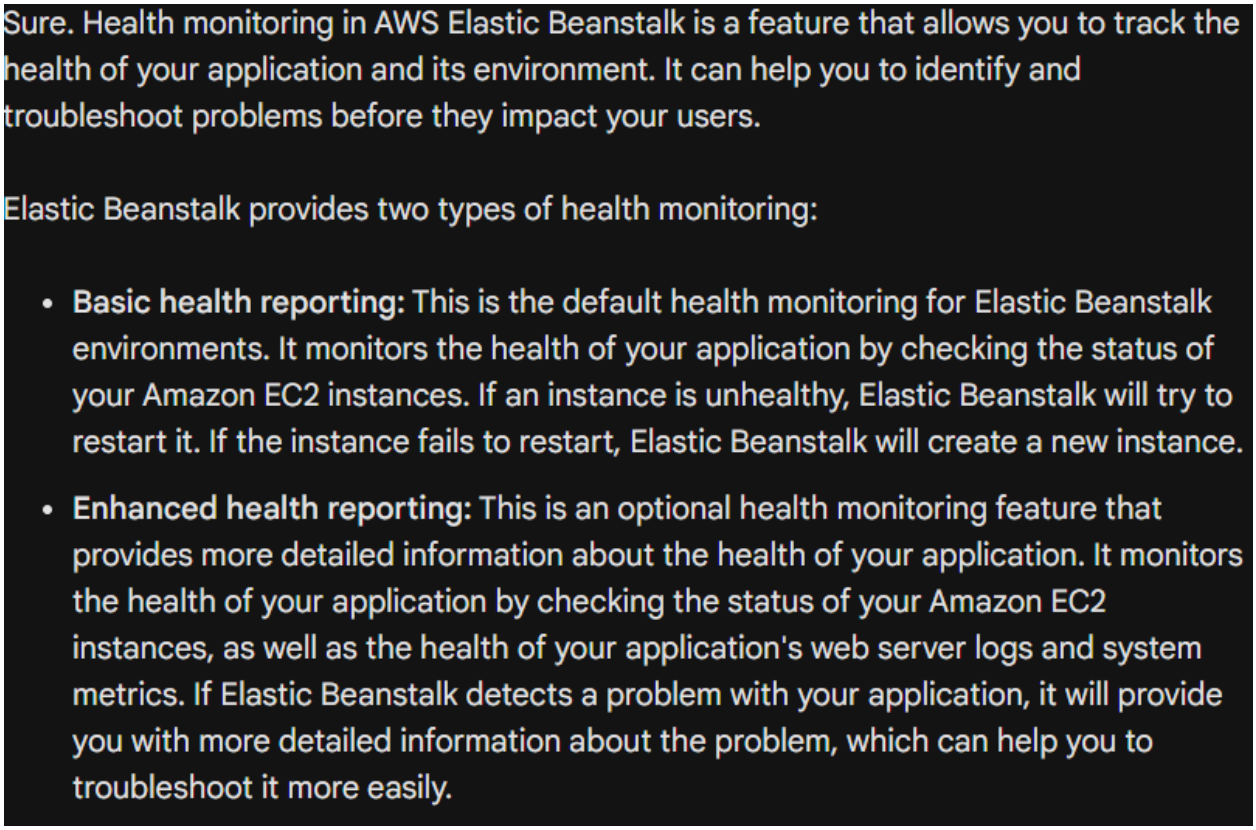
- Health agent pushes metrics to CloudWatch
- Checks for app health, publishes health events







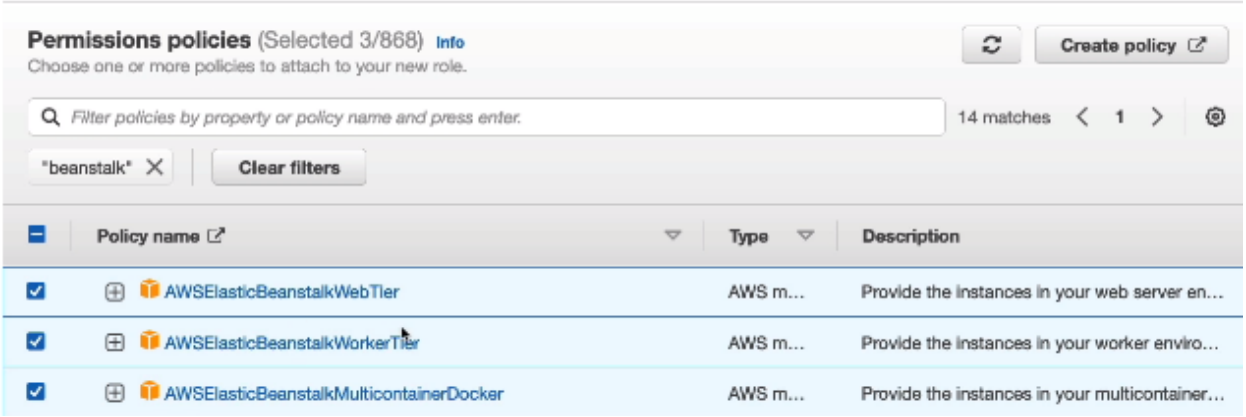
18.

19. 

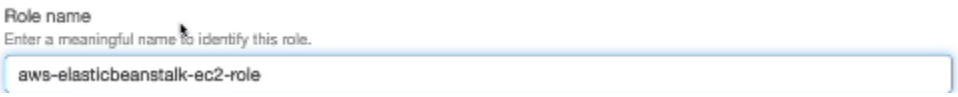
BeanStalk HandsOn:

1. First we go to aws elastic beanstalk service ..and click on create application
2. Then we give our applicationname and environment name…
3. Domain will be given automatically
4. Next we choose nodejs as our managed platform …then..if have our own code we can upload it in application..or else we can use sample application
5. Next we have to choose instances..we choose single instance which is free

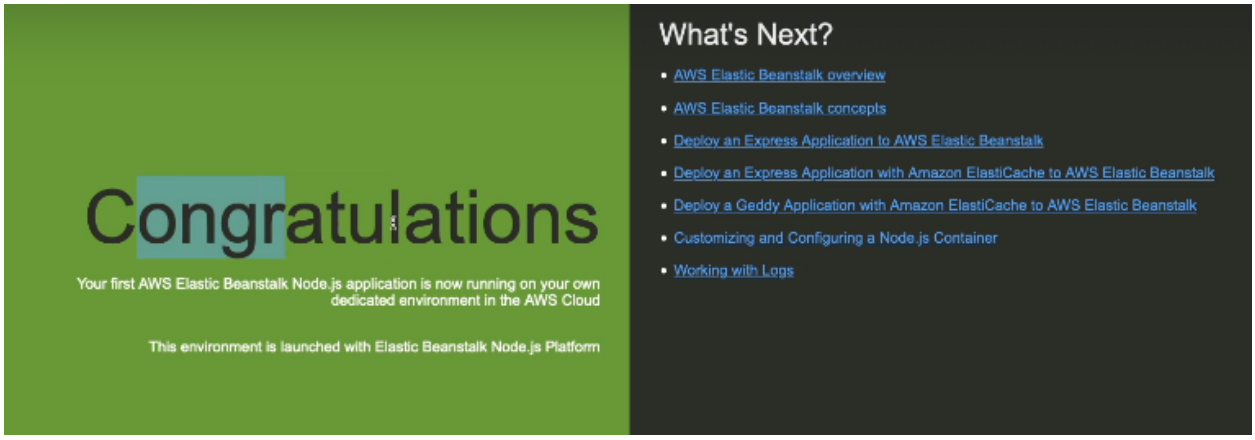
6. Then we choose a service..here we are creating a new service ..for that we have to create a role…then we go into IAM->roles->create role->select these policies



7. Next we give the role name

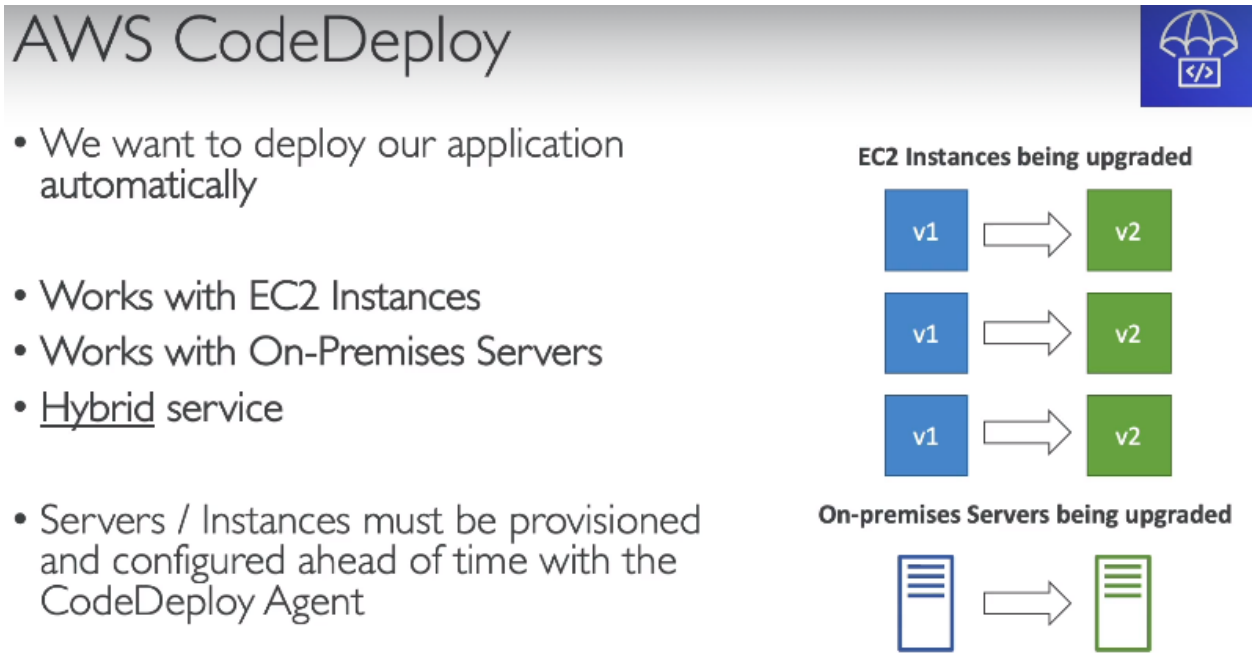


8. We do this IAM and all…if we didnt get ec2 instance profile ..prefilled..while creating beanstalk
9. As we got ec2 instance prefilled for our application..we don't need to create roles
10. Next we skip to review our application..then click on submit
11. This will create our first beanstalk
12. but the idea is that all the resources that Beanstalk wants to create are actually created behind the scenes by CloudFormation.
13. In the backend our beanstalk has created an ec2instance,Elastic ip , Auto scaling grps
14. After successfully launching environment ..if we click on domain name we get



15. And also we can create a second application..
16. In the end we deleted the our application

AWS CodeDeploy

1. 

2. Sure. AWS CodeDeploy is a service that automates the deployment of applications to Amazon EC2 instances, on-premises instances, serverless Lambda functions, or Amazon ECS services. It can help you to:

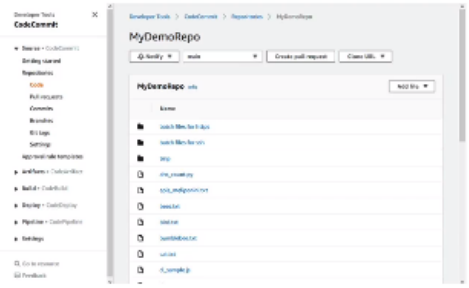
   - **Deploy your application to multiple instances in parallel.** CodeDeploy can deploy your application to multiple instances in parallel, which can help you to reduce the deployment time.
   - **Roll back deployments if there are problems.** If there are problems with a deployment, CodeDeploy can roll back the deployment to the previous version of your application.
   - **Monitor the deployment process.** CodeDeploy can monitor the deployment process and provide you with detailed information about the status of the deployment.
   - **Automate deployments with CodeDeploy Pipelines.** CodeDeploy Pipelines is a feature that allows you to automate the deployment process by defining a series of steps that CodeDeploy will execute. This can help you to streamline the deployment process and reduce the risk of errors.

AWS CodeCommit

1.

AWS CodeCommit

- Before pushing the application code to servers, it needs to be stored somewhere
- Developers usually store **code in a repository, using the Git technology**
- A famous public offering is GitHub, AWS' competing product is **CodeCommit**
- CodeCommit:
  - Source-control service that **hosts Git-based repositories**
  - Makes it easy to **collaborate with others on code**
  - The code changes are automatically **versioned**
- Benefits:
  - Fully managed
  - Scalable & highly available
  - Private, Secured, Integrated with AWS



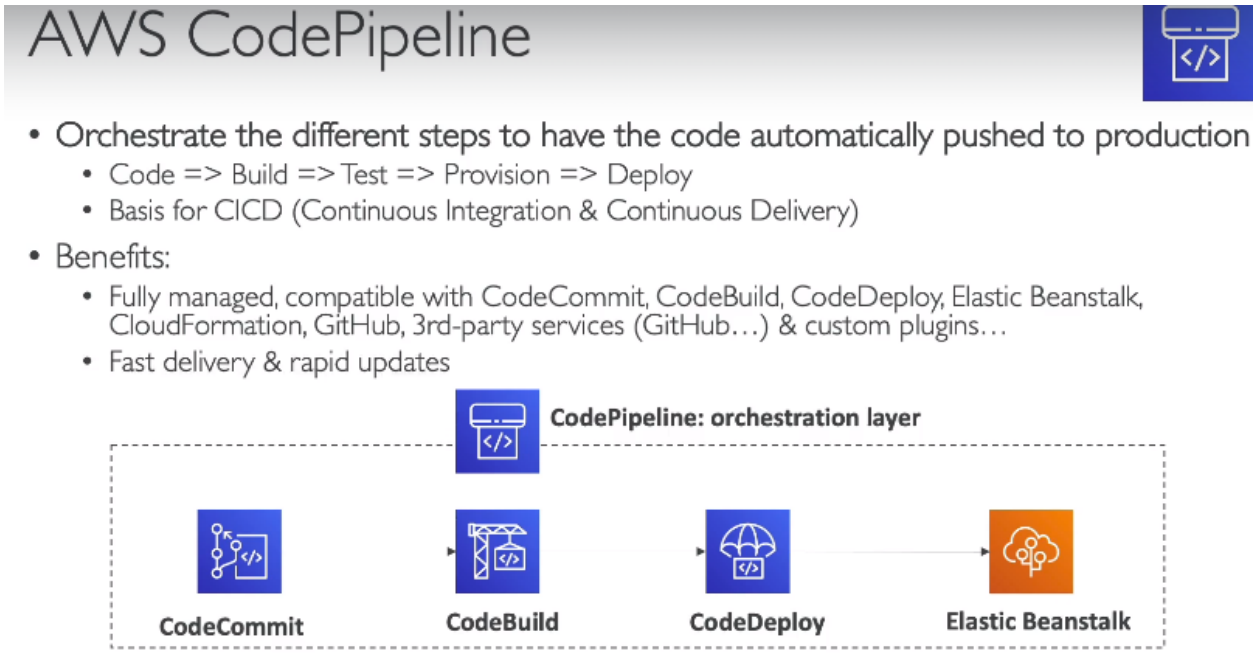
2. AWS CodeCommit is a fully-managed source control service that makes it easy for you to store and manage your code in the cloud. It offers a number of features that make it a good choice for a variety of use cases, including:

   - **Collaboration:** CodeCommit is designed for collaborative software development. It allows you to create branches and pull requests, which can help you to track changes to your code and collaborate with others on your team.
   - **Encryption:** CodeCommit encrypts your code at rest and in transit. This helps to protect your code from unauthorized access.
   - **Access control:** CodeCommit uses IAM roles and users to control who has access to your repositories. This helps you to secure your code and prevent unauthorized access.
   - **High availability:** CodeCommit is a highly available service. Your repositories are replicated across multiple Availability Zones, so you can be sure that your code is always available.
   - **Durability:** CodeCommit is a durable service. Your repositories are stored in Amazon S3, which is a highly durable storage service.
   - **Scalability:** CodeCommit is a scalable service. You can easily add or remove repositories as needed.

AWS CodeBuild

1.
2. If we have our code in codecommit ..then codebuild will retrieve the code..and it will build the code to be ready to be deployed artifact

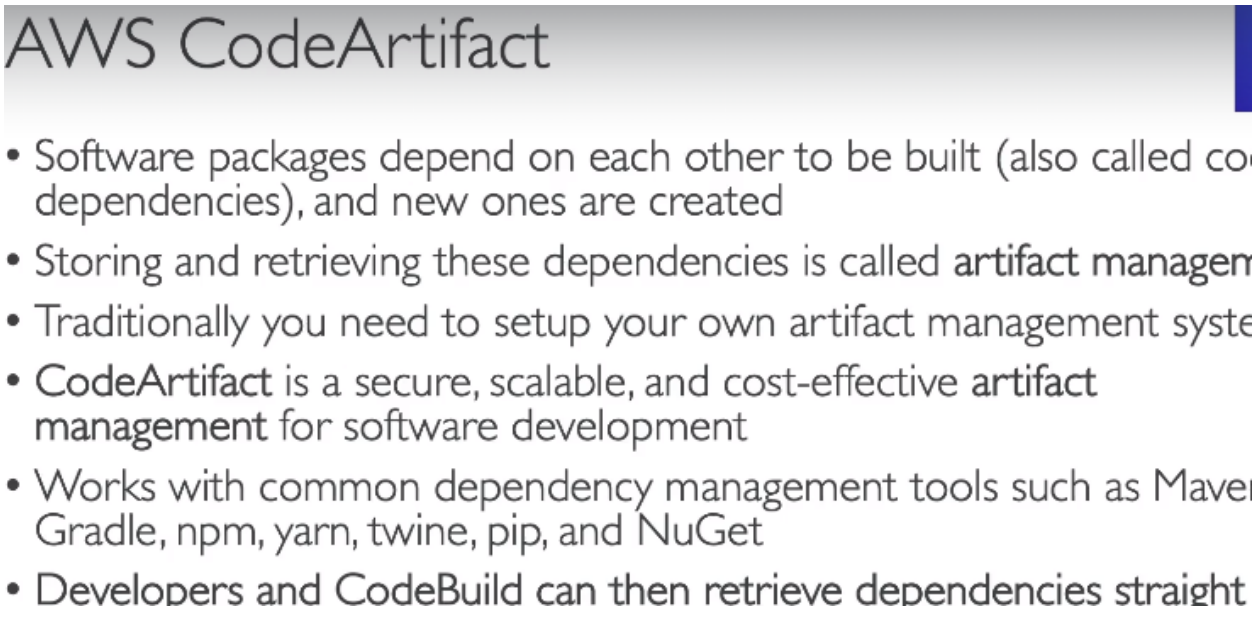
AWS CodePipeline

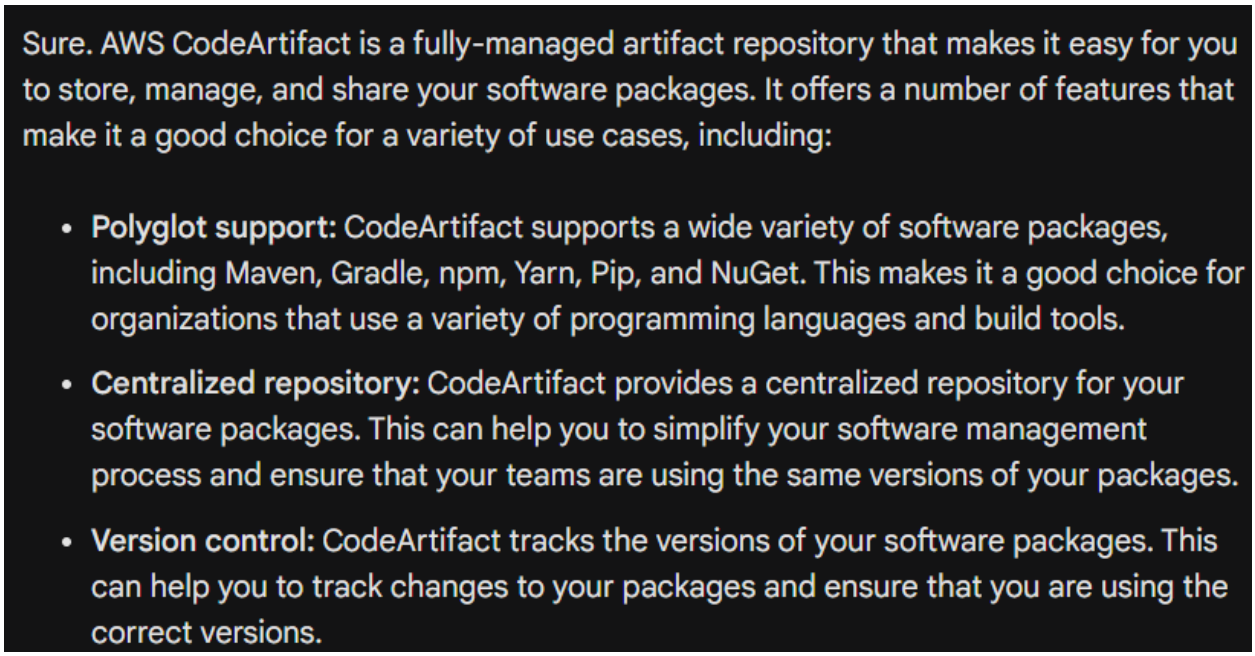


1.
2. It is like pipeline..
3. It orchestrates the layer
4. it will take its code from CodeCommit

5. build it with CodeBuild, then decide to deploy it with CodeDeploy, and may be deployed into an Elastic Beanstalk environment as one example. But it's just one way of building a pipeline,
6. And so anytime you see orchestration of pipeline in your exam, you have to think AWS CodePipeline.

AWS CodeArtifact

1. 

2. 

3. Here are some of the specific use cases for AWS CodeArtifact:

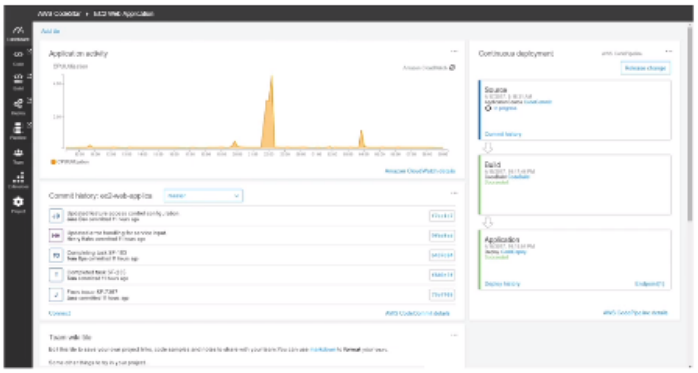
   - **Store and manage software packages:** CodeArtifact can be used to store and manage all of your software packages, including dependencies, plugins, and tools.
   - **Share software packages:** CodeArtifact can be used to share software packages with your team members or with the public.
   - **Track software package versions:** CodeArtifact can be used to track the versions of your software packages. This can help you to ensure that you are using the correct versions of your packages.
   - **Audit software package access:** CodeArtifact can be used to audit who has accessed your software packages and when they accessed them. This can help you to track down security vulnerabilities or compliance issues.
   - **Deploy software packages:** CodeArtifact can be used to deploy software packages to your development, staging, and production environments. This can help you to automate the deployment process and ensure that your software packages are always deployed in a consistent manner.

4. So from an exam perspective, CodeArtifact is going to be very helpful if your team needs a artifact management system or a place to store their code dependencies.

AWS CodeStar

AWS CodeStar

- Unified UI to easily manage software development activities in one place
- "Quick way" to get started to correctly set-up CodeCommit, CodePipeline, CodeBuild, CodeDeploy, Elastic Beanstalk, EC2, etc…
- Can edit the code "in-the-cloud" using AWS Cloud9



1.

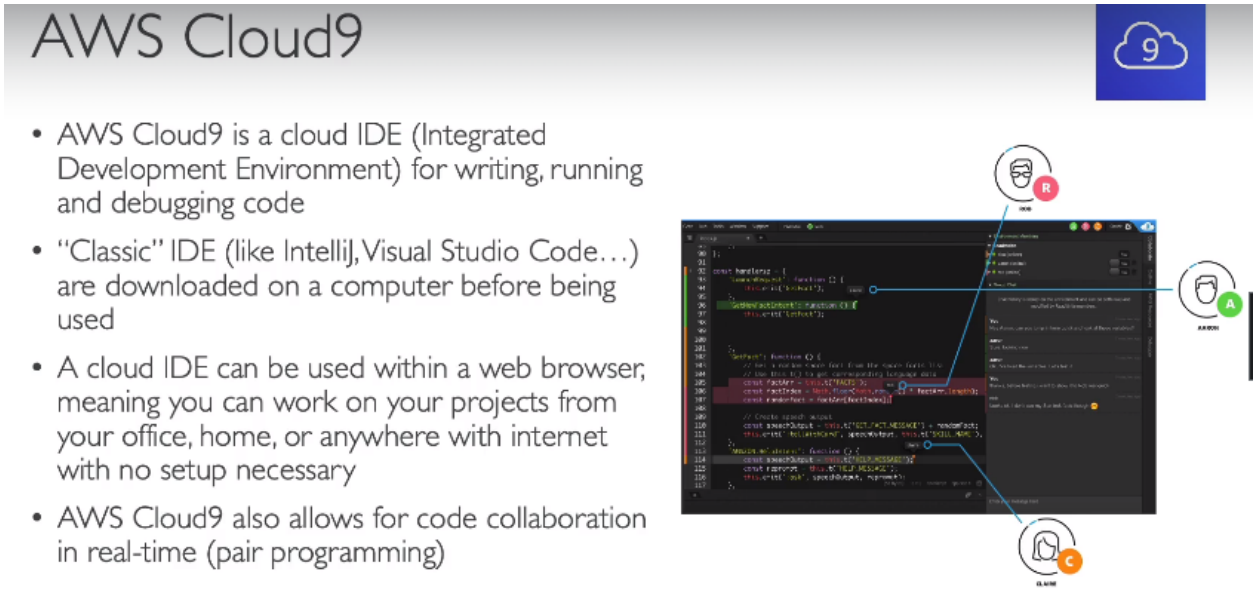
2. AWS CodeStar is a service that helps you quickly and easily create, build, and deploy applications on AWS. It provides a unified experience for managing your software development projects, including:

   - **Project templates:** CodeStar provides a variety of project templates that you can use to get started quickly. These templates include everything you need to build a complete application, including source code, build scripts, and deployment instructions.
   - **Integrated development environment (IDE):** CodeStar integrates with a variety of IDEs, so you can use your favorite IDE to develop your applications.
   - **Continuous integration and continuous delivery (CI/CD):** CodeStar includes a CI/CD pipeline that you can use to automate the testing, deployment, and monitoring of your applications.
   - **Issue tracking:** CodeStar integrates with Atlassian JIRA, so you can track issues and bugs in your projects.
   - **Collaboration:** CodeStar provides a centralized dashboard where you can collaborate with your team members on your projects.

3. So all of this makes CodeStar a central servicethat allows you developers to quickly start with development while using the best CI/CD practices, and that's all you need to remember going into the exam.

AWS Cloud9

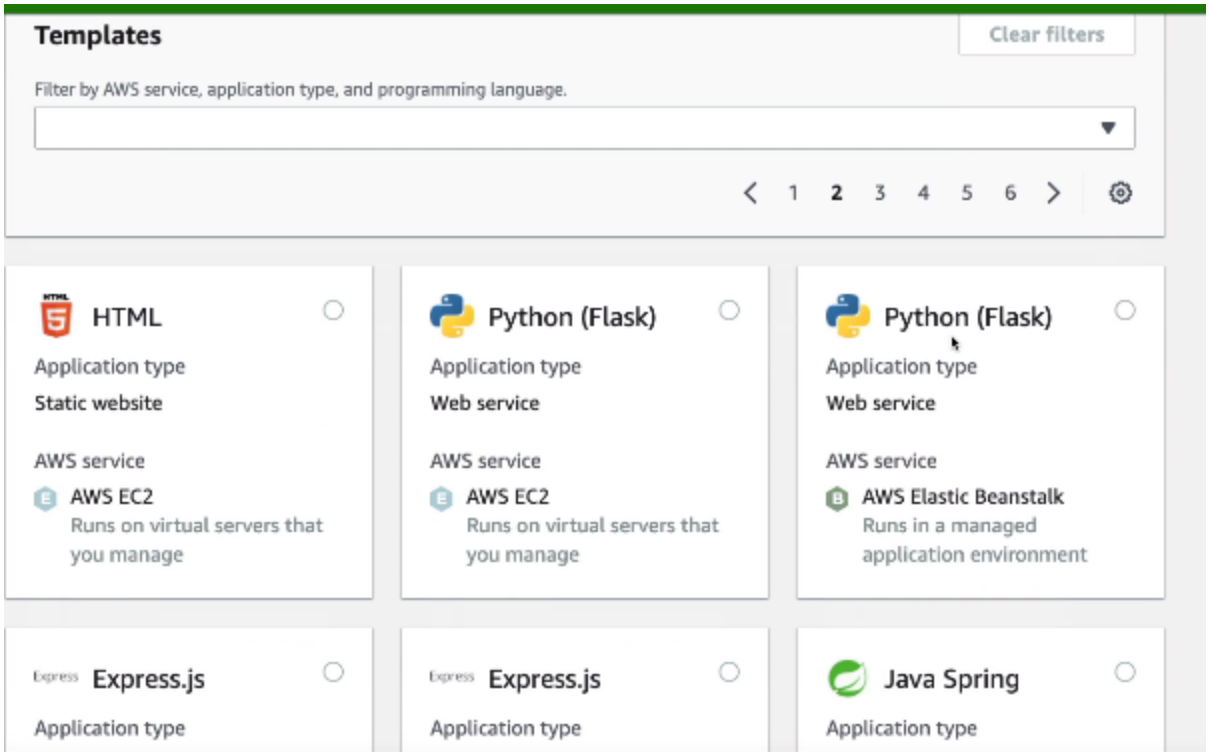
1. So Cloud9 is a cloud IDE, which stands for integrated development environment, that is used for writing, running, and debugging code directly in the cloud. So it looks like a code editor, but it is run in the cloud, so it is run in a web browser.

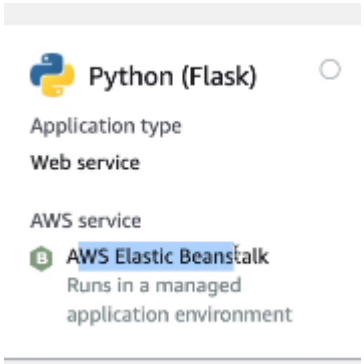


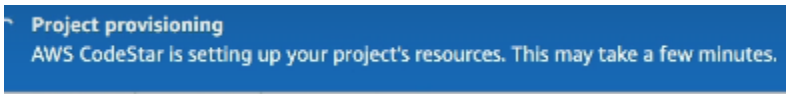
2.
3. Here we can see..there are 3 people working on the same code..

CodeStar and Cloud9 Hands on:

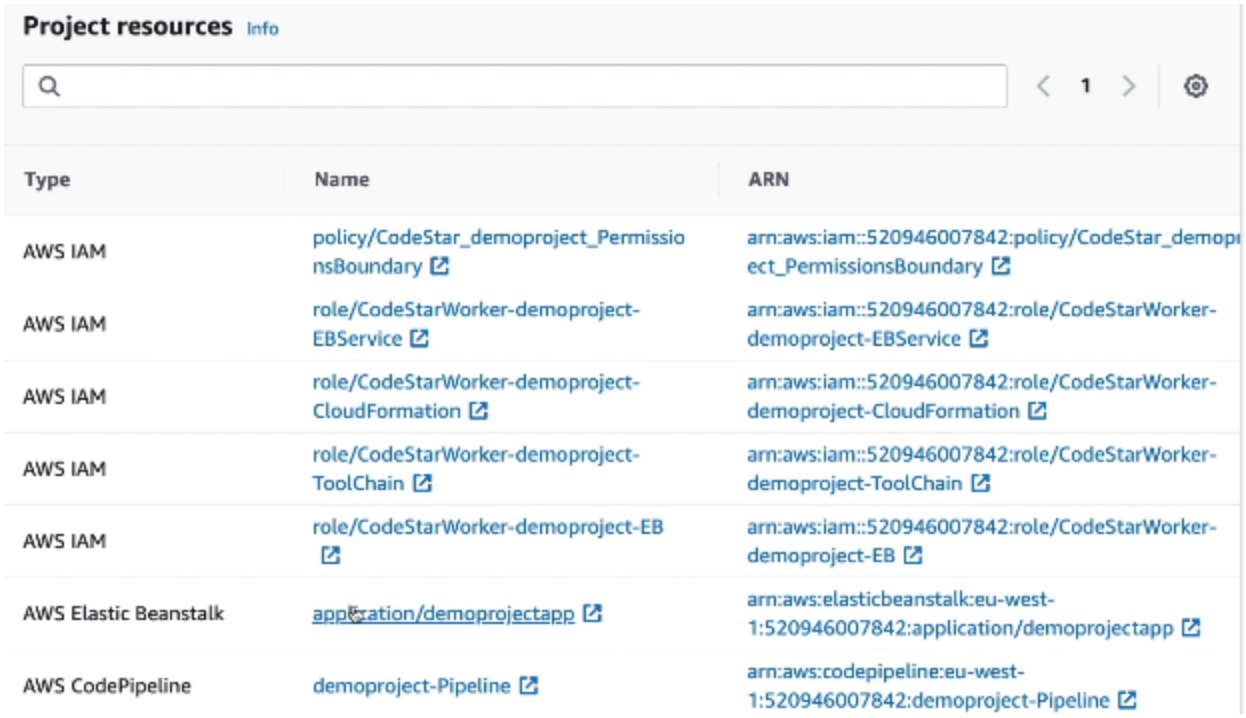
1. Here first we will go to codecommit
2. As we can see, we have CodeCommit. We have CodeArtifacts, CodeBuild, CodeDeploy, and CodePipeline available from the left hand side as standalone tools.
3. In the codecommit..we can create a new repository..But for now, what I want to show you is how we can get started with all of them at once, using only one service. And that service is CodeStar.
4. In codestar first we will create a project..and next we have to choose a template

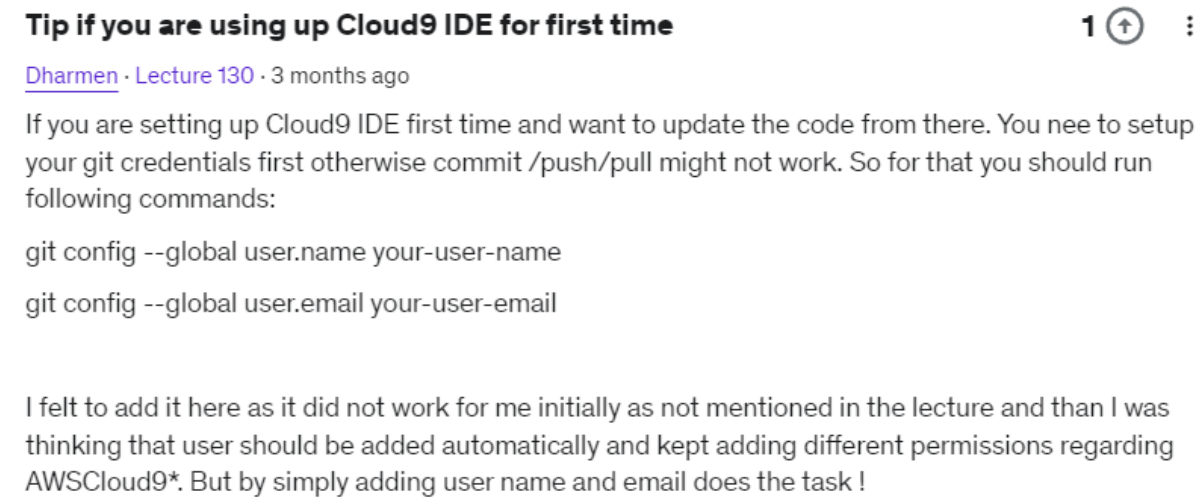
5. 



6. We choose this template
7. Next we give project name ..and it will also create a repository for our project and an ec2 instance..next we have to give key-pair ..and next click on create project
8. 

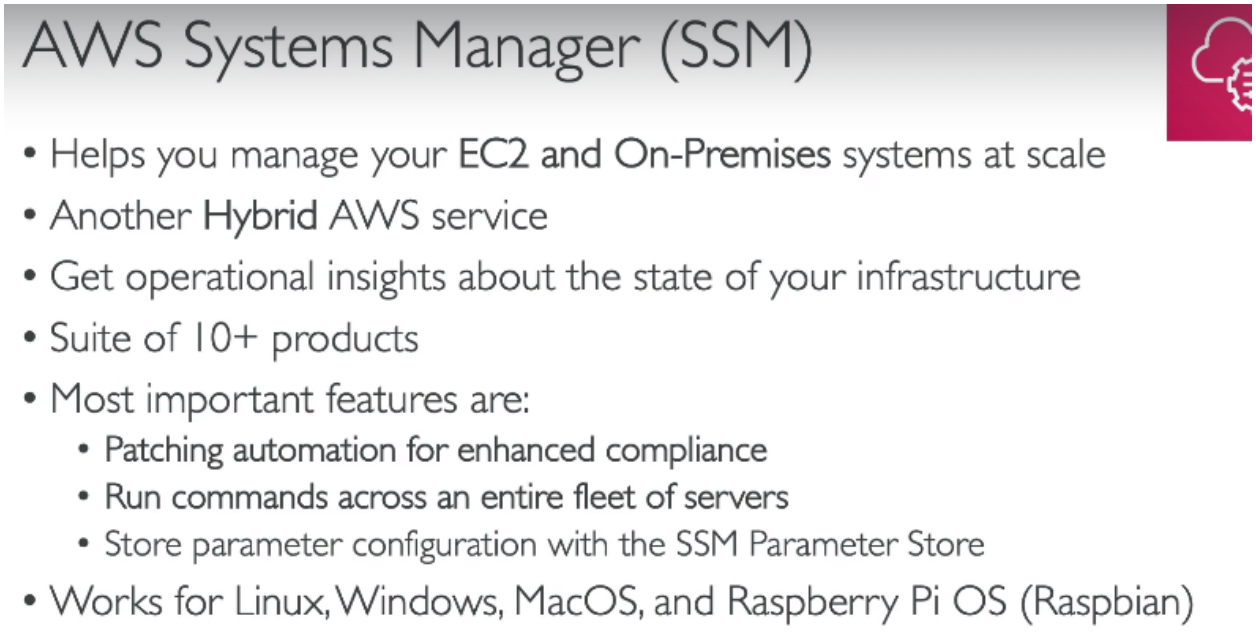
9. After the setup…if we go to project activity..we can see there's an pipeline created for our project
10. It also created lots of project resources for us

11. 

and many more
12. If we go to code pipeline..we can see a pipeline that codestar created for our project
13. But more importantly, if we go to Beanstalk, and in Beanstalk we can see the application. If I click on this URL right here, it takes me into the project itself, which has been deployed on Beanstalk. And it says, "Output. Hello World," which shows that the application is working `{"Output": "Hello World"}`
14. Next we are going to create cloud9 env for our project
15. After the cloud9 env is ready..if we click on open IDE..we get the cloud9 IDE...this allows us to run or edit our code..directly in the cloud
16. 

17. Later we have modified the hello world to hello k..and committed the changes
18. After committing if we push our code..
19. We can see all the commits in codestar repository
20. It also triggers the pipeline
21. So it really shows you the end-to-end workflow of a developer within AWS.
22. In the end we del our cloud9 env and also we will del codestar project…to cleanup

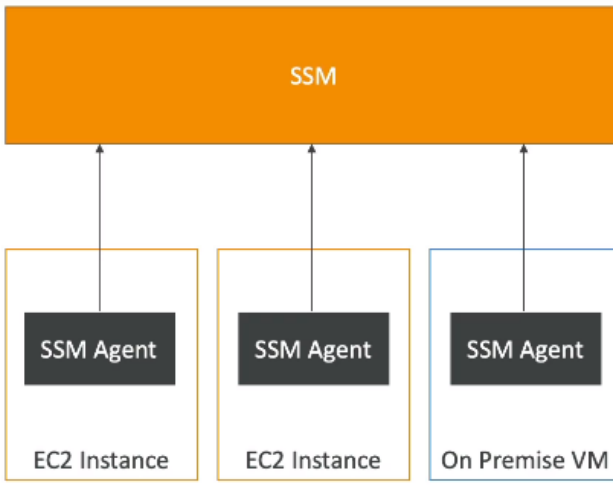
AWS Systems Manager(SSM)



1.
2. So, from an exam perspective, anytime you see a way to patch your fleet of EC2 instances or On-Premises servers,
3. you have to think about SSM, or if you wanted to run a command consistently across all your servers, again, SSM would be the right way.

How Systems Manager works

- We need to install the SSM agent onto the systems we control
- Installed by default on Amazon Linux AMI & some Ubuntu AMI
- If an instance can't be controlled with SSM, it's probably an issue with the SSM agent!
- Thanks to the SSM agent, we can **run commands, patch & configure** our servers



4.

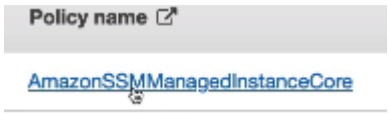
SSM Session Manager

Systems Manager – SSM Session Manager

- Allows you to start a secure shell on your EC2 and on-premises servers
- No SSH access, bastion hosts, or SSH keys needed
- No port 22 needed (better security)

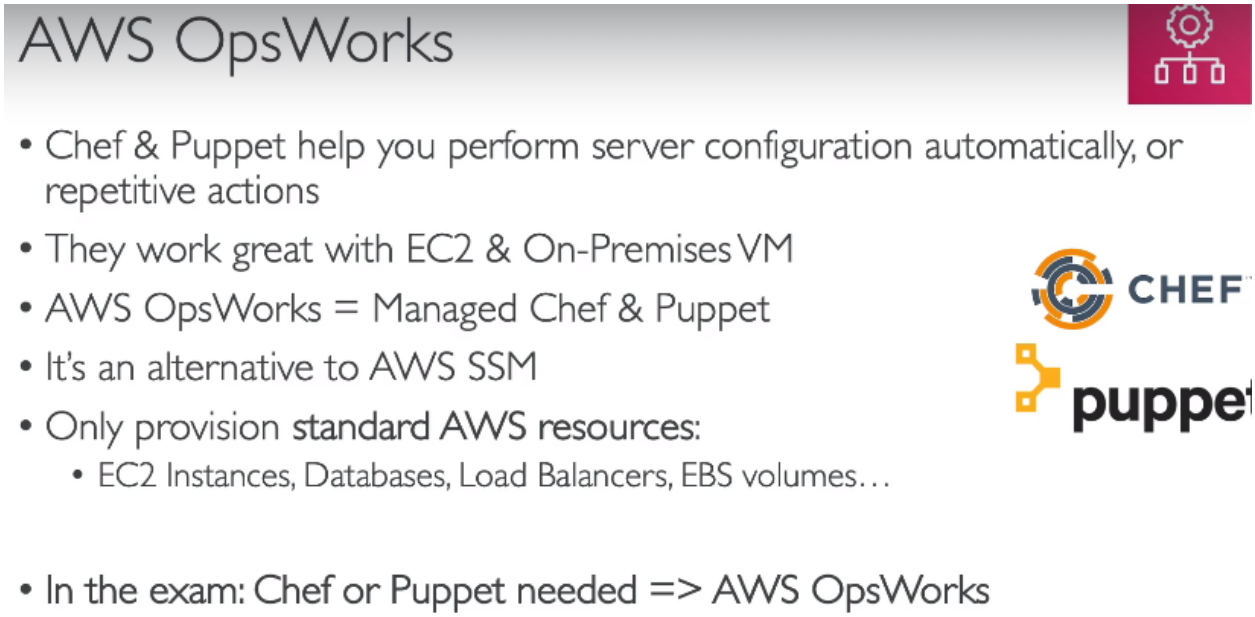
1.
2. Here our ec2 instance has ssm agent and the agent is connected to the Session Manager Service
3. It will make more sense if we do hands on

4. Here first we have launch our ec2 instance..while creating an instance… what I need to do actually is to attach an IAM instance profile to my instance to allow it to talk to the SSM service.
5. So we create new IAM profile and create a role and attach this policy to talk with SSM service



6. After the role is created ..we select it in IAM instance profile..which is demoSSH…
7. Next we create this instance…after creating it..we go to SSM manager
8. There on left dashboard ..we click on fleet manager
9. Fleet Manager is a service where all the EC2 instances that are registered with SSM will appear here. So they're called managed nodes.
10. Next click on session manager …and we will start a session on our ec2 instance ..here we can access secure shell..without SSH
11. So that means that using the SSM secure shell, we're able to have indeed a secure shell directly from AWS without having SSH security keys and SSH access.
12. So we have 3 ways to access our ec2 instance
13. Number one is to open the port 22 and then use SSH keys and with a terminal to do the SSH command.
14. Number two is to use EC2 Instance Connect and that didn't require to get SSH keys because they will be temporarily uploaded onto the Amazon EC2 instance if we need to.
15. And then we've explored the third option which is Session Manager. So we need to make sure that we had an EC2 instance with Amazon Linux 2 and make sure that this EC2 instance had an IAM role
16. and this IAM role, okay, as you can see under security, had an IAM role and this IAM role allowed access from the EC2 instance to Systems Manager and this is what allowed us to get the secure shell running.

AWS OpsWorks



- In the exam: Chef or Puppet needed => AWS OpsWorks

1.
2. So from an exam perspective,
3. OpsWorks is very simple to reason about.
4. Any time you see Chef or Puppet that is needed, you need to think about OpsWorks, and this is going to be the extent of what is required from you from the exam.

Deployment Summary

Deployment - Summary

- CloudFormation: (AWS only)
  - Infrastructure as Code, works with almost all of AWS resources
  - Repeat across Regions & Accounts
- Beanstalk: (AWS only)
  - Platform as a Service (PaaS), limited to certain programming languages or Docker
  - Deploy code consistently with a known architecture: ex, ALB + EC2 + RDS
- CodeDeploy (hybrid): deploy & upgrade any application onto servers
- Systems Manager (hybrid): patch, configure and run commands at scale
- OpsWorks (hybrid): managed Chef and Puppet in AWS

1.

Developer Services - Summary

- CodeCommit: Store code in private git repository (version controlled)
- CodeBuild: Build & test code in AWS
- CodeDeploy: Deploy code onto servers
- CodePipeline: Orchestration of pipeline (from code to build to deploy)
- CodeArtifact: Store software packages / dependencies on AWS
- CodeStar: Unified view for allowing developers to do CICD and code
- Cloud9: Cloud IDE (Integrated Development Environment) with collab

2.

- AWS CDK: Define your cloud infrastructure using a programming language

3.