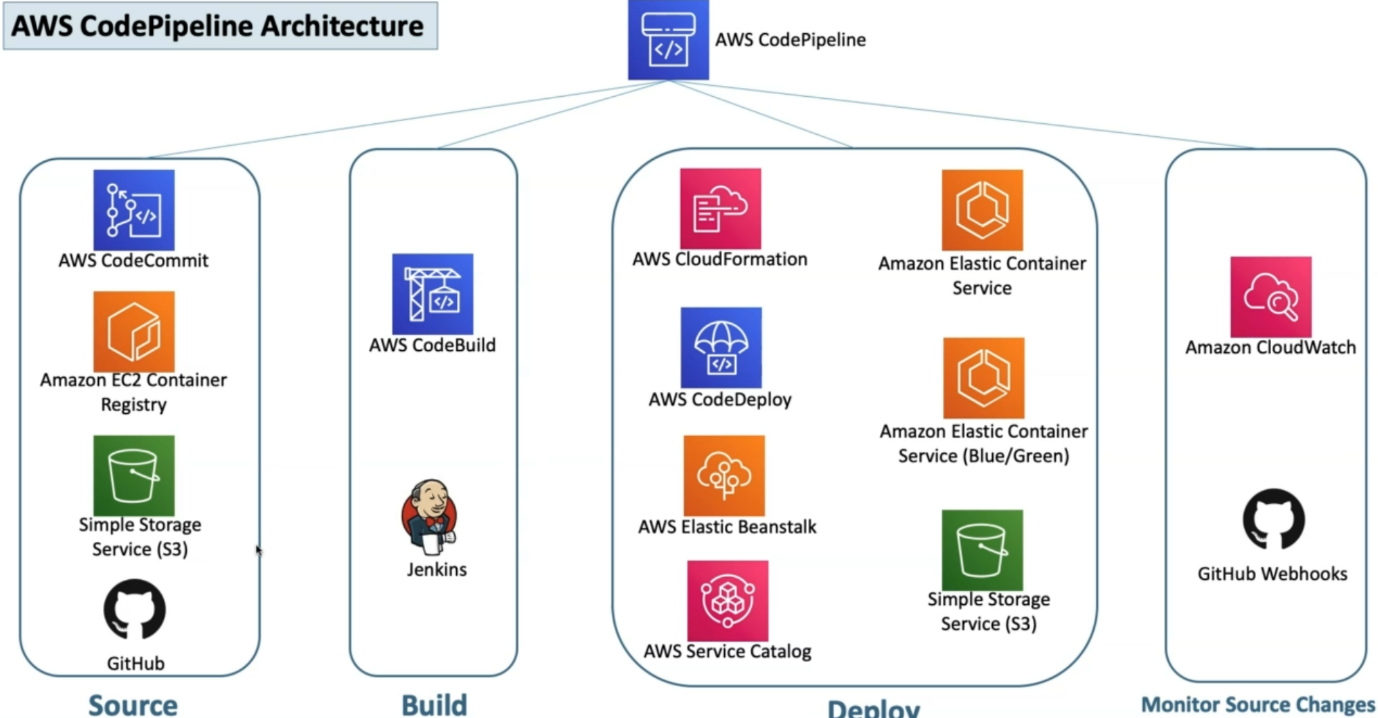
**08. CodePipeline – Introduction**



**AWS codepipeline architecture**



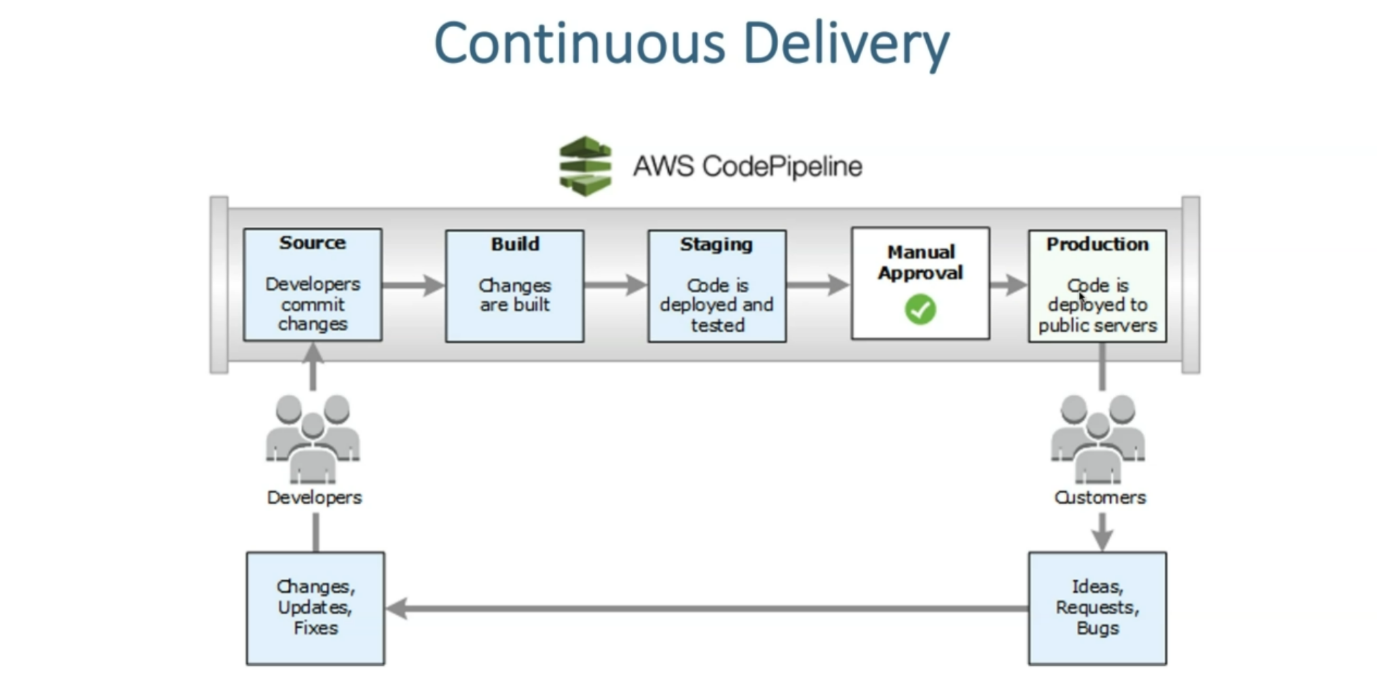
--- we already know that source, build and deploy are the major stages inside the codepipeline.

--- So, from source perspective, it primarily depends or it primarily integrated with code commit EC2 container registry and then simple storage service and also github.

--- In addition to that. from Bill perspective, it uses Codebuild and jenkins and from deploy perspective, it uses cloud formation, cloud deploy, elastic beanstack, service catalog, simple storage service, elastic container service and also elastic container service blue green deployments.

--- to monitor the source. It is going to use either the cloud watch when we are using our other services and then we are using GitHub. It is going to use the GitHub webbooks.

**what is this continuous delivery?**

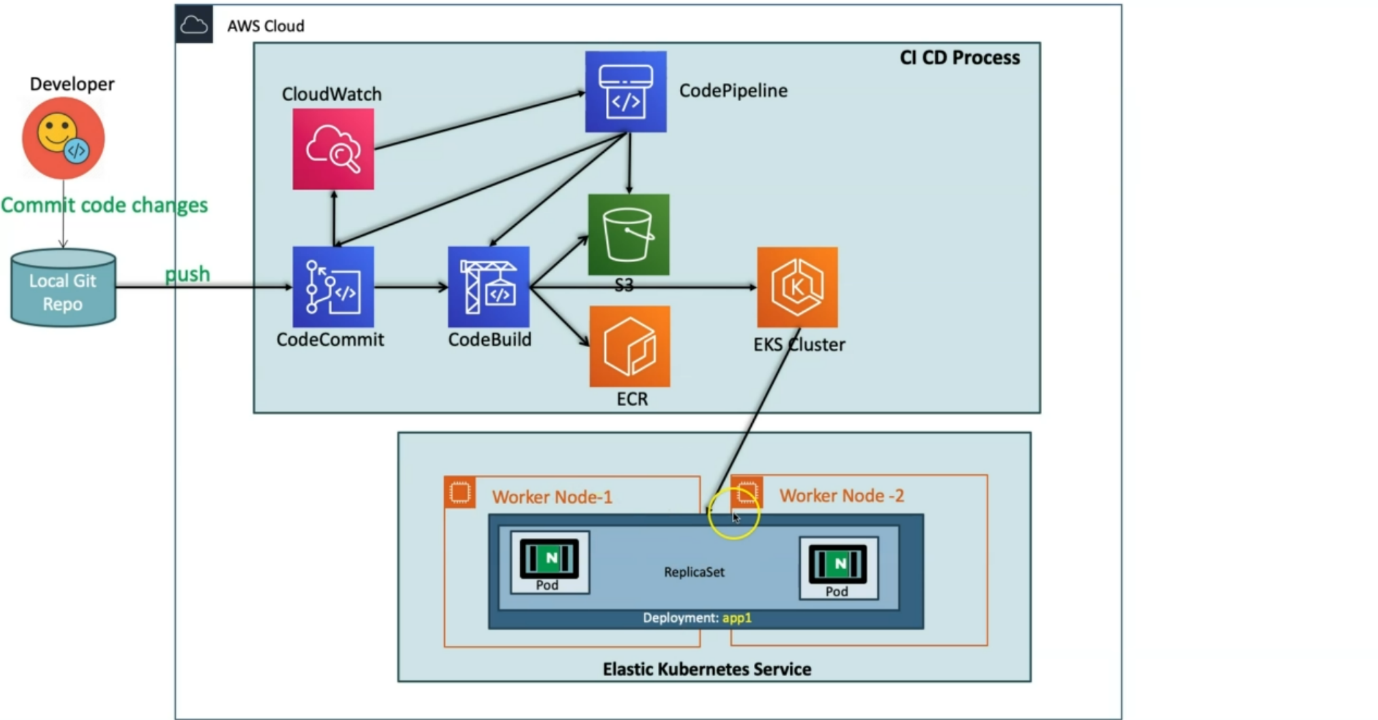


--- So, as a developer, I check in the code to the source and what happens at that time. whenever the change happened on our source, a cloud watch event is triggered and the entire pipeline gets triggered.

--- the build process starts and then changes are built and then pushed to the S3 bucket for our artifacts and then the same is pulled and then deployed to our respective staging environment.

--- an email approval Flow is sent and then respective approver, if he approves. Then only it will be deployed for production.

**Continuous delivery flow implementation**



--- as a dollar power, I will do my Docker files and also Kube manifests and I'll check into the local git repo and will push the same to the codecommit repository.

--- What happens…? immediately a cloud watch event will be triggered, if I configure the pipeline and then pipeline is available for this respective source and pipeline will be triggered. immediately the codebuild Process will start and then whatever we define in a codebuild process, which is nothing but buildspec.yml.

--- the entire things are going to get executed then it will create a new docker image and then it will push the docker image to the docker Repository. It is nothing but a elastic container repository and it also pushes the artifacts to the S3 bucket.

--- we will apply kubectl -f Kube-manifests. it will take into consideration, whenever this command is executed, automatically changed docker image will be applied to our respective deployment and then all pods will get terminated and then new pods will get create.