**Benefits of Kubernetes for CD/CD:**

The key to a successful CI/CD pipeline is to ensure that the application updates occur in a swift and automated manner. Kubernetes offers plenty of solutions for common problems that programmers face throughout this process.

1. **Ability to Containerize the Code:** With the platform, we can run the apps in containers. This ensures they have the resources and libraries necessary, while also preventing common issues that arise between library versions and application components. Containerizing the code makes the app portable between environments, while also making them easy to replicate and scale.
2. **Orchestrate Deployment with the Platform:** Kubernetes makes the deployment process easier in a number of ways. Running apps on containers doesn’t solve every problem in the CI/CD pipeline as you still need to manage these apps. The platform can do everything from deploying them to monitoring their health and scaling them to meet customer demand.
3. Reducing the Time of Release Cycles: Many programmers struggle when they stick to a manual testing and deployment process. Doing so can cause delays, which push back the production timeline. A manual CI/CD process leads to more code-merge collisions, as well as the time that customers have to wait for patches and updates.
4. **Solving Outages:** A manual infrastructure management process also causes headaches for coding teams because someone has to remain alert in case any outages happen. A number of issues can arise such as a power outage or an unforeseen traffic spike beyond capacity. This requires someone to be alert around the clock to solve these issues. If the app is down, ORG will lose money and customers. With an automated platform such as Kubernetes, we can automate patches and updates to solve these outages.

**Procedure:**

* Making use of Kubernetes Continuous Deploy Plugin : <https://wiki.jenkins.io/display/JENKINS/Kubernetes+Continuous+Deploy+Plugin>
* **Jenkins Automation:**
* Jenkins provides easy-to-setup workflow to automate the deployments. With Pipeline support, it is flexible to build the zero-downtime deployment workflow, and visualize the deployment steps. To facilitate the deployment process for Kubernetes resources, there is Kubernetes Continuous Deploy and the Azure Container Service plugins built based on the kubernetes-client. We can deploy the resource to Azure Kubernetes Service (AKS) or the general Kubernetes clusters without the need of kubectl, and it supports variable substitution in the resource configuration so we can deploy environment-specific resources to the clusters without updating the resource config. An example Jenkins Pipeline to demonstrate the blue/green deployment to target platform. The flow is like the following:

1. Pre-clean: clean workspace.
2. SCM: pulling code from the source control management system.
3. Prepare Image: prepare the application docker images and upload them to some Docker repository.
4. Check Env: determine the active and inactive environment, which drives the following deployment.
5. Deploy: deploy the new application resource configuration to the inactive environment. With the Azure Container Service plugin, this can be done with: acsDeploy azureCredentialsId: 'stored-azure-credentials-id', configFilePaths: "glob/path/to/\*/resource-config-\*.yml", containerService: "aks-name | AKS", resourceGroupName: "resource-group-name", enableConfigSubstitution: true
6. Verify Staged: verify the deployment to the inactive environment to ensure it is working properly. Again, note this is in the production environment, so be careful not to pollute live application data during tests.
7. Confirm: Optionally, send email notifications for manual user approval to proceed with the actual environment switch.
8. Switch: Switch the frontend service endpoint routing to the inactive environment. This is just another service deployment to the AKS Kubernetes cluster.
9. Verify Prod: verify the frontend service endpoint is working properly with the new environment.
10. Post-clean: do some post clean on the temporary files.
11. For the Rolling Update strategy, simply deploy the deployment configuration to the Kubernetes cluster, which is a simple, single step.

* Blue/green Deployment strategy.