10 Most Important Steps To Troubleshoot ArgoCD Issues

Kirill Kazakov
Senior DevOps Engineer

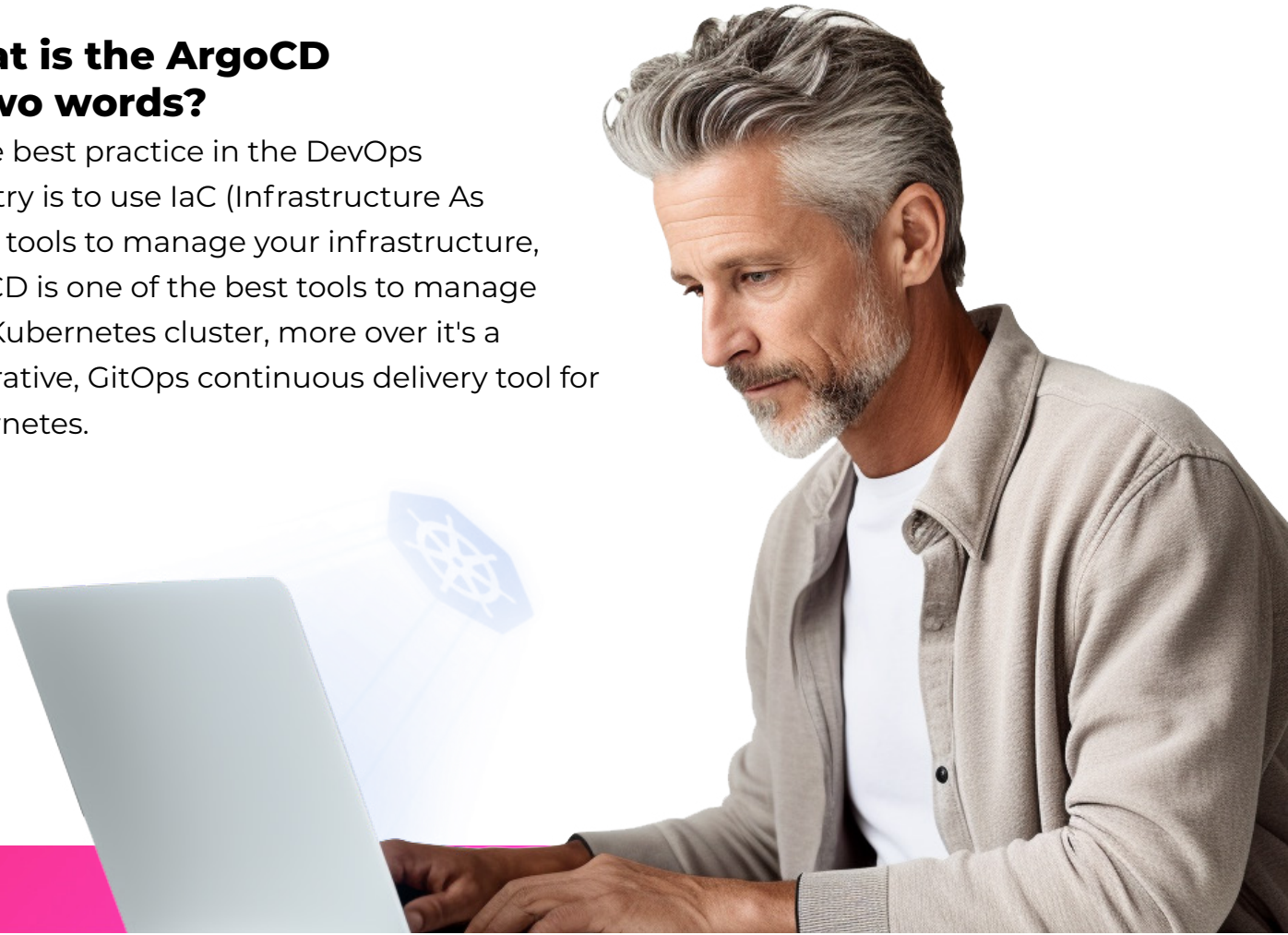
Introduction

If you are reading this article, you are probably already familiar with ArgoCD.

If not, we recommend you to read the official documentation first.
Beside huge benefits ArgoCD brings an additional layer of the complexity of the CI/CD process, ArgoCD has a lot of moving parts, and it's not always easy to troubleshoot the issues. In this article, we will try to cover the most common issues that you can face during the ArgoCD installation, configuration and management process.

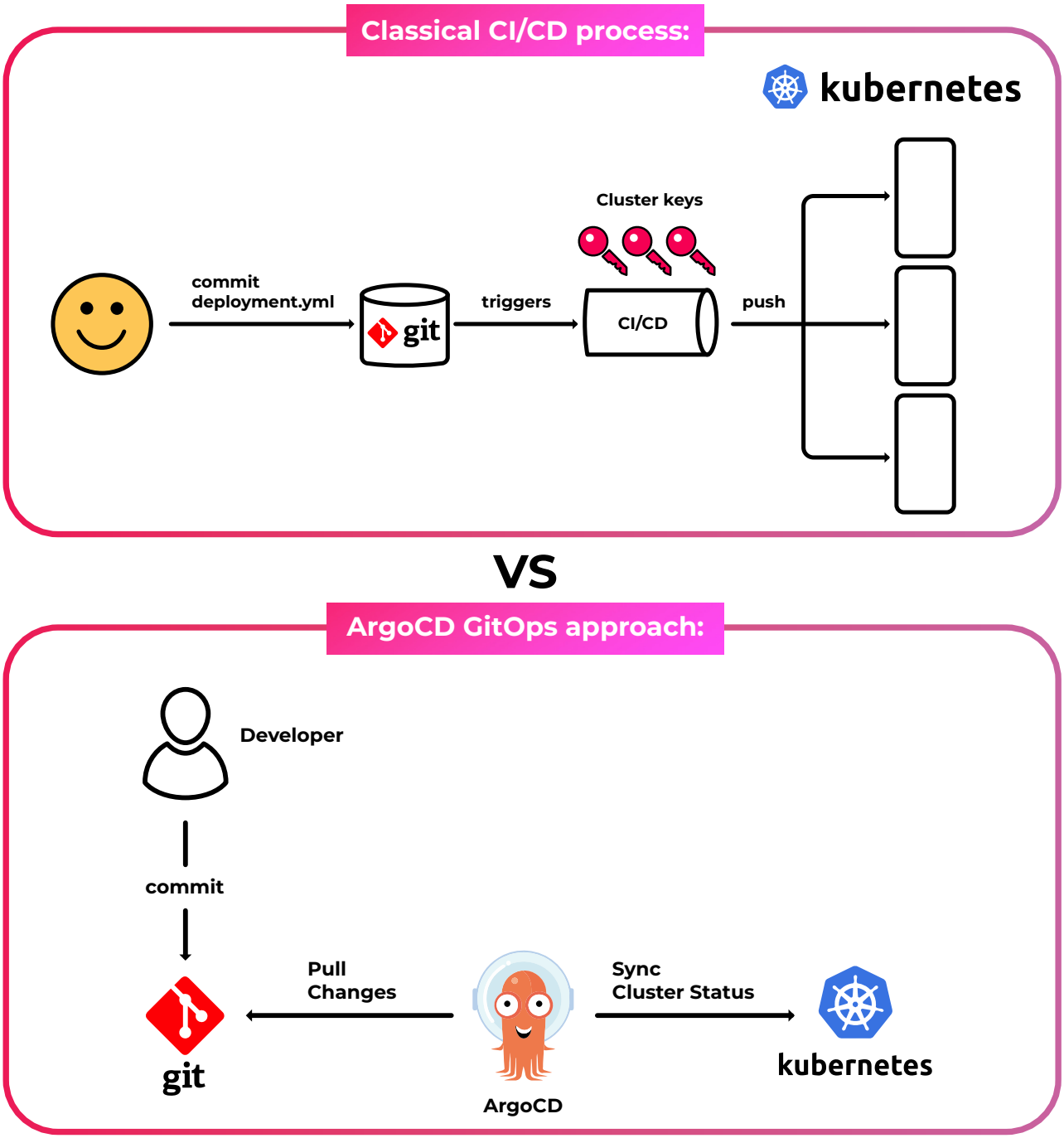
What is the ArgoCD in two words?

As the best practice in the DevOps industry is to use IaC (Infrastructure As Code) tools to manage your infrastructure, ArgoCD is one of the best tools to manage your Kubernetes cluster, more over it's a declarative, GitOps continuous delivery tool for Kubernetes.



How does it work under the hood?

ArgoCD it's k8s manifests in your cluster, and it's a controller that continuously monitors running applications for changes to the manifest that it tracks (e.g. Deployment, Service, ConfigMap, etc.) and compares the changes to the desired state stored in the Git repository. If there are any differences, ArgoCD will perform a kubectl application to sync the cluster state with the desired state.
The difference between a classical CI/CD process and ArgoCD illustrate on the picture below.



We guess you already installed the ArgoCD (we prefer to use ah Helm way) and played with it, so let's move to the next section.

Cases

1. What is the default password of argocd?

Context: You just installed a fresh ArgoCD and want to login to the UI, but you don't know the default password.

If you used the Helm chart to install the ArgoCD, you can find the default password in the k8s secrets. By default an ArgoCD setup admin user, which has full access to the ArgoCD. The default password for the admin user is auto generated during the installation process and stored in the k8s secret.

Solution: Get the password:

```
kubectl get secret argocd-initial-admin-secret -n argocd -o jsonpath="{.data.password}" | base64 -d
```

Update password:

```
argocd login <ARGOCD_SERVER>
argocd account update-password
```

Read More

2. How to manage users and groups in ArgoCD?

Context: To split users and roles between the different teams, you need to create different users and groups in ArgoCD.

Solution:
To achieve this you can create new local users via configmap\ArgocdCli or use an external authentication provider like LDAP, SAML, OIDC, etc. We prefer the last one. In this way you can manage users and groups in the external authentication provider and sync them with ArgoCD.
ArgoCD supports a huge pack of integrations like LDAP, SAML, OIDC, etc.
As part of installation ArgoCD uses DEX provider, which is a lightweight OpenID Connect identity provider (IdP) with a focus on Kubernetes and OpenShift clusters.
List of the connectors you can find here.
Read more about an ArgoCD user management here.

Example to configure ArgoCD SSO with GitHub (via helm values):

```
## Argo Configs
 configs:
 cm:
 # Dex configuration
 dex.config: |
  connectors:
   # GitHub example
  - type: github
  id: github
  name: GitHub
  config:
  clientID: aabbccddeeff00112233
  clientSecret: $dex.github.clientSecret # Alternatively $<some_K8S_secret>:dex.
github.clientSecret
  orgs:
- name: your-github-org
```

Read More

3. Where can we find inspiration for the implementation of ArgoCD for advanced cases?

Context: You want to implement ArgoCD for your project, but you don't know how to start and what are the best practices.

Solution:
We recommend to check the official ArgoCD examples:

- A base git repo with examples.
- More examples with ApplicationSet.

Example of the ArgoCD ApplicationSet to use Pull Request Generator. This fits well with the style of building a test environment when you create a pull request.

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
 name: myapps
spec:
 generators:
 - pullRequest:
  github:
   # The GitHub organization or user.
   owner: myorg
   # The Github repository
   repo: myrepository
   # For GitHub Enterprise (optional)
   api: https://git.example.com/
   # Reference to a Secret containing an access token. (optional)
   tokenRef:
    secretName: github-token
    key: token
   # (optional) use a GitHub App to access the API instead of a PAT.
   appSecretName: github-app-repo-creds
   # Labels are used to filter the PRs that you want to target. (optional)
   labels:
   - preview
 requeueAfterSeconds: 1800
template:
# ...
```

Full example.
Read more.

4. Delete ArgoCD Application without deleting the resources

Context: You want to delete the ArgoCD Application, but you don't want to delete the resources.

Solution: You can use the kubectl delete command with the --cascade=false flag.

```
kubectl delete application <APP_NAME> --cascade=false
```

In this case ArgoCD does not track anymore your resources, but they will still be in the cluster. It could be useful, when you need to debug smth, or fix it. Don't worry, after you will apply the next time the ArgoCD Application, the resources will be synced with the desired state.

5. ArgoCD application stuck at deleting

Context: You want to delete the ArgoCD Application, but it's stuck at deleting.
There are a lot of reasons why it could happen, but the most common is that you have a finalizer in your ArgoCD Application, which prevents the deletion of the ArgoCD Application.

Solution: You can use the kubectl patch command to remove the finalizer from the ArgoCD Application.

```
kubectl patch application <APP_NAME> -p '{"metadata":{"finalizers": []}}' --type=merge
```

Do not forget to disable auto-sync for this ArgoCD Application before you will remove the finalizer.
Additional tips:

- Check the logs of the ArgoCD Application controller
- Check namespace of the ArgoCD Application
- Check the permissions of the ArgoCD Application controller for this namespace

6. Disable ArgoCD Application auto-sync

Context: You want to disable ArgoCD auto-sync for specific Applications.
It's pretty easy to achieve this, you can use ArgoCD UI or CLI to disable auto-sync for specific Applications.
When do we need it? Most popular case is when you need to debug smth in the cluster, and you don't want to sync your resources with the desired state in the git repository.

Solution: Use argocd CLI to disable auto-sync for specific Applications.

```
argocd app set <APP_NAME> --sync-policy none
```

7. How to Disable ArgoCD auto-sync in the App of Apps pattern or ApplicationSet?

Context: You want to disable ArgoCD auto-sync for debug purposes, but you have a lot of Applications in the App of Apps pattern or ApplicationSet which override the auto-sync policy (from the previous step 6).

Solution for the App of APPs: You need first to disable aut-sync for the root Application, and then for Application that you need.

```
argocd app set <ROOT_APP_NAME> --sync-policy none
argocd app set <APP_NAME> --sync-policy none
```

Solution for the ApplicationSet way: You need to disable auto-sync for the ApplicationSet, and then for Application that you need.

```
argocd appset set <APPSET_NAME> --sync-policy none
argocd app set <APP_NAME> --sync-policy none
```

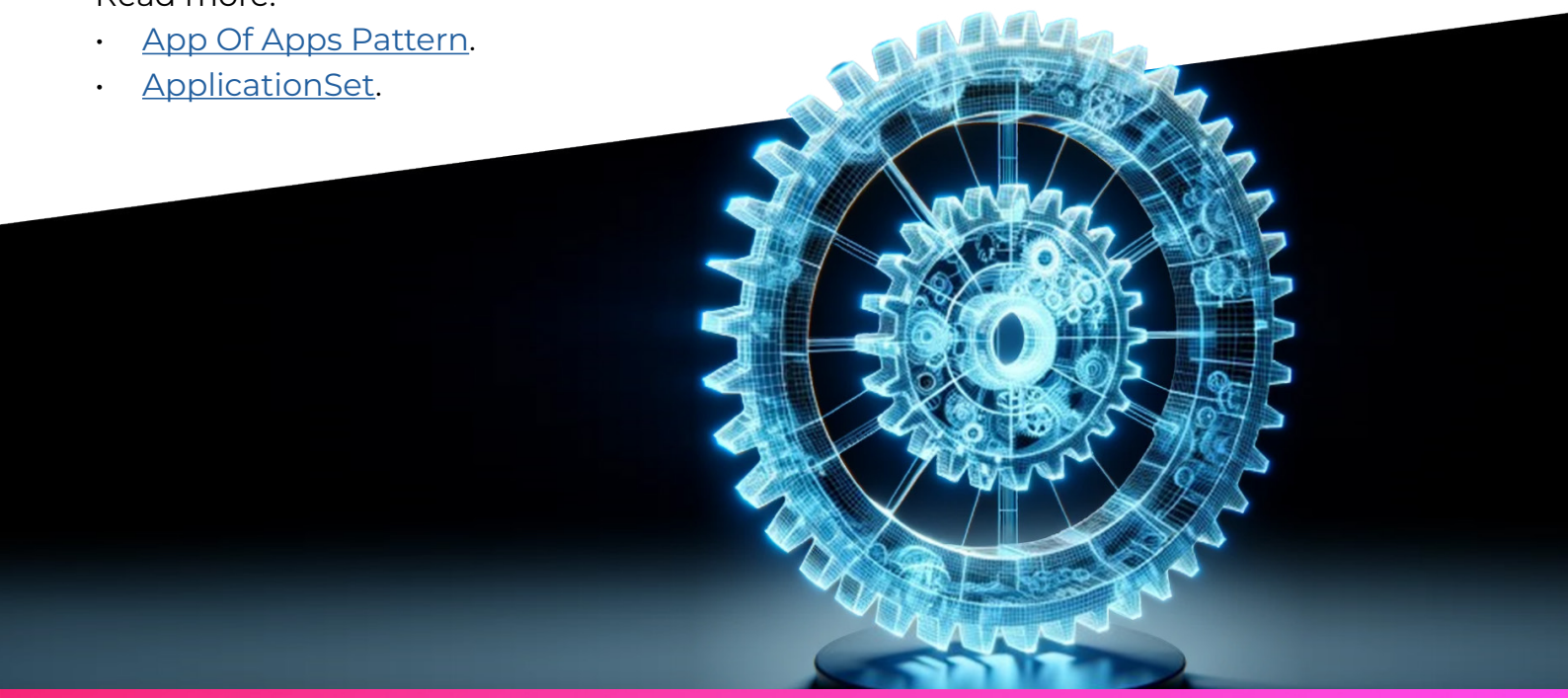
Quick solution for both: Delete the root Application or ApplicationSet with cascade=false flag, and then path sync settings for the target app:

```
kubectl delete application <ROOT_APP_NAME> --cascade=false
argocd app set <APP_NAME> --sync-policy none
```

After you finish with debugging, just reapply your Application or ApplicationSet, and ArgoCD will sync your resources with the desired state in the git repository.

Read more:

- App Of Apps Pattern.
- ApplicationSet.



8. How to ignore ArgoCD sync policy for specific k8s resources?

Context: You want to control some k8s resources with some other tools, and you don't want to sync them with ArgoCD, but one of the resources is a part of the ArgoCD Application.
The most popular case is when you want to use the Kubernetes Horizontal Pod Autoscaler or KEDA autoscaler to scale your application, but you don't want to sync the HPA resource with ArgoCD.
If you don't apply this workaround, ArgoCD will override your HPA resource with the desired state in the git repository, then HPA will override it again, and so on continuously.

Solution: You can use the JsonPatch to ignore the specific k8s resources in the ArgoCD Application.

spec:

```
ignoreDifferences:
- group: apps
  kind: Deployment
  jqPathExpressions:
  - .spec.template.spec.initContainers[] | select(.name == "injected-init-container")
```

Read More

9. Application not showing in ArgoCD, when you use ApplicationSet

Context: You use ApplicationSet to generate ArgoCD Applications, but you don't see them (Applications) in the ArgoCD UI or in the CLI.
The main thing you need to understand, that ApplicationSet is a controller, which generates ArgoCD Applications based on the template. Keep in mind you can't see ApplicationSet objects in the UI. The most popular case is that there is some error in the template, and ApplicationSet can't generate ArgoCD Applications.

Solution: You need to check the logs of the ArgoCD Application controller:

```
kubectl logs -n argocd deployment/argocd-application-controller
```

Also, you can check the status of the ApplicationSet:

```
kubectl get appset <APPSET_NAME> -o yaml
```

10. Logs and events

Context: A key of most of the issues is in the logs and events. You need to check them first to understand what is going on.
ArgoCD has a lot of components, and each of them has its own logs and events. You need to check them all to understand what is going on. For example, if you have an issue with the ArgoCD Application, you need to check the logs of the ArgoCD Application controller, and the logs of the ArgoCD Application itself.
Or if you have an issue with ApplicationSet, you need to check the logs of the ArgoCD Application controller, and the logs of the ArgoCD ApplicationSet controller.

Solution for Application: You can check the logs of the ArgoCD Application controller:

```
kubectl logs -n argocd deployment/argocd-application-controller
```

Solution for ApplicationSet: You can check the logs of the ArgoCD ApplicationSet controller:

```
kubectl logs-n argocd deployment/argocd-applicationset-controller
```

Useful links:

- ArgoCD FAQ
- Getting Started