Azure WebApp Deployment: Key Concepts and Troubleshooting

1. What is a Deployment Slot?

A **deployment slot** in Azure Web Apps allows you to create multiple environments (e.g., staging, testing, and production) within the same App Service. Each slot has its own:

- Unique URL
- App settings and connection strings
- Deployment history

Benefits of Deployment Slots:

- **Zero-downtime deployments:** Swap new versions without affecting users.
- **Testing before production:** Validate changes in a staging slot before promoting.
- **Rollback support:** Easily revert to a previous stable version.
- **Configuration isolation:** Each slot can have different environment settings.

Common Deployment Slots:

- **Production** (default slot)
- **Staging** (for pre-production testing)
- **Custom slots** (e.g., development, QA, etc.)

---

2. Why is Application Deployment Failing in Azure WebApp?

Deployment failures in Azure Web Apps can be caused by several factors. Below are common reasons and troubleshooting steps:

a. Incorrect Configuration

- Missing or incorrect **app settings** and **connection strings**.
- Environment variables not set properly.

Solution:

- Check and update settings in **Azure Portal → App Service → Configuration**.
- Use az webapp config appsettings list -n <app-name> -g <resource-group> to review settings.

b. Deployment Errors

- Issues with **CI/CD pipelines** (e.g., GitHub Actions, Azure DevOps, Jenkins).
- Failed deployments due to incorrect startup scripts.

Solution:

- Check logs in **Azure Portal → Deployment Center**.
- Run az webapp log tail -n <app-name> -g <resource-group> to view real-time logs.

- Ensure your pipeline has the correct **build and release** configurations.

c. Code or Dependency Issues

- Missing dependencies in requirements.txt (Python) or package.json (Node.js).
- Mismatched .NET runtime versions.

Solution:

- Check logs under **Azure Portal → App Service Logs**.
- Use Kudu Console (https://<app-name>.scm.azurewebsites.net) to manually debug.
- Run az webapp show -n <app-name> -g <resource-group> --query "siteConfig" to verify runtime settings.

d. Deployment Conflicts

- Another deployment is already in progress.
- A locked process prevents updates.

Solution:

- Restart the web app using:
- az webapp restart -n <app-name> -g <resource-group>
- Use the **Deployment Center** to cancel active deployments.

---

3. How to Undo the Deployment of an Azure WebApp?

If a deployment causes issues, you can revert to a previous working version.

a. Swap Deployment Slots

If using deployment slots:

1. Go to **Azure Portal → App Service → Deployment Slots**.
2. Select the **staging slot** and click **Swap with Production**.
3. This will rollback the production environment to the previous stable version.

CLI Command:

az webapp deployment slot swap -n <app-name> -g <resource-group> --slot staging --target-slot production

b. Redeploy a Previous Version

- If using **Azure DevOps or GitHub Actions**, redeploy a previous commit.
- If using ZIP deployment, re-upload an older version:
- az webapp deployment source config-zip -g <resource-group> -n <app-name> --src old_version.zip

c. Restore from Backup

- If **backups are enabled**, restore the app from a previous backup:
- az webapp restore -g <resource-group> -n <app-name> --backup-name <backup-id>
- Alternatively, use **Azure Portal → Backups → Restore**.

Conclusion

Deployment slots provide a safe way to test and roll back deployments without downtime. Understanding the common causes of deployment failures and knowing how to revert changes can help ensure a smooth deployment process for Azure Web Apps.