DAVID BYERS

STOP DEVELOPING ON PRODUCTION!

CI/CD workflows for small business that are fast, effective, and inexpensive.

WHO AM I?

- David Byers
- ColdFusion Developer for 26 years
- Held multiple CTO and fractional CTO positions
- Code Monkey Studios
- Proof that if you come to CF Summit enough, they eventually let you speak.

WHAT WE WILL COVER TODAY

- The Problem: Why CI/CD and Why Now?
- Let’s Get on the Same Page
- Repository Platforms, CI/CD Tools and Pipelines: Automating Deployments Without Breaking the Bank
- Environments: Stop Developing on Production
- Deployment Strategies: Ship Like a Pro
- Some basic CI/CD examples

THE PROBLEM: WHY CI/CD AND WHY NOW?

QUICK POLL

MY PREDICTIONS:

~80% WILL HAVE DEVELOPED AGAINST PRODUCTION AT SOME POINT.

~25% WILL BE DEVELOPING AGAINST PRODUCTION NOW.

COMMON DEVELOPMENT SINS

Why CI/CD is no longer optional.

EDITING CODE DIRECTLY ON PRODUCTION SERVERS.

COMMON DEVELOPMENT SINS

Why CI/CD is no longer optional.

FTP'ING FILES DIRECTLY (AND HOPING NOTHING BREAKS.)

COMMON DEVELOPMENT SINS

Why CI/CD is no longer optional.

NO VERSION CONTROL, OR USING GIT LIKE A FILING CABINET.

COMMON DEVELOPMENT SINS

Why CI/CD is no longer optional.

NO BACKUPS. NO ROLLBACK PLAN.

COMMON DEVELOPMENT SINS

Why CI/CD is no longer optional.

“TEST IT LIVE” VERSUS USING A DEVELOPMENT OR STAGING SERVER.

COMMON DEVELOPMENT SINS

Why CI/CD is no longer optional.

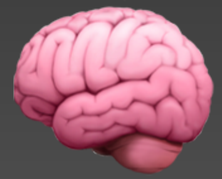
SINGLE SOURCE OF KNOWLEDGE.
ONLY ONE PERSON KNOWS HOW IT ALL WORKS,
...AND THEY'RE ON VACATION. 

COMMON DEVELOPMENT SINS

Why CI/CD is no longer optional.

DANGEROUS PROCESSES

LACK OF SAFETY NETS



KNOWLEDGE SILOS

WHAT THESE THINGS CAUSE:

- Broken sites, inconvenient emergencies, and late-night fire drills
- Customer complaints, lack of faith in the business.
- No audit trail of changes or who did what.
- Fear of touching working code. (“It’s fragile. Don’t breathe on it.”)
- Lost time and trust.
- Paralysis: Features delayed because deployment is terrifying.

WHY?

REASONS COMPANIES GIVE

~~REASONS~~ EXCUSES COMPANIES GIVE

- “We’re too small for pipelines.”
- “CI/CD is for big teams with big budgets.”
- “We don’t have time to set all that up.”
- “It’s just easier to edit live…”
- *(Sound familiar?)*

LET'S GET ON THE SAME PAGE

LET'S GET ON THE SAME PAGE

What is CI/CD?

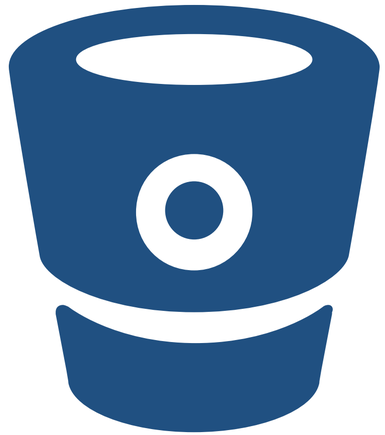
- CI: Continuous Integration. Automatic testing and integrating code every time it's committed to your repository.
- CD: Continuous Development / Deployment / Delivery. Automation to move code from your repo to a staging or production server without manual steps.
- CI/CD isn't just for big companies or fancy apps. It's a repeatable way to stop editing code directly on production.

LET'S GET ON THE SAME PAGE

What is a repository?

- A repository or repo is your **source of truth** for your code.
- It stores your app's code, history, and changes collaboratively and safely.
- SaaS hosted Git repositories



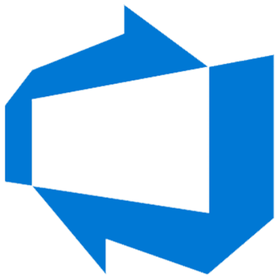


LET'S GET ON THE SAME PAGE

What is a repository?

- A repository or repo is your **source of truth** for your code.
- It stores your app's code, history, and changes collaboratively and safely.
- Platform-native repositories for specific ecosystems.

Google Cloud Platform

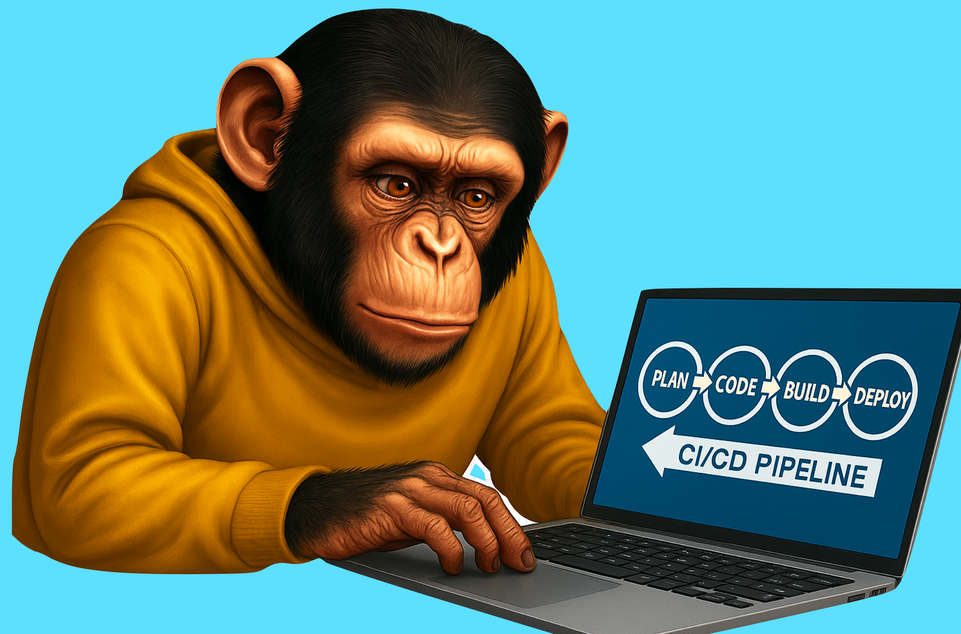


Azure DevOps

LET'S GET ON THE SAME PAGE

What is a Pipeline?

- A set of repeatable, automated steps that run when you push code.
- Example:
  - Push to Main →
    - Run unit tests →
    - Build docker image →
    - Deploy to staging server →
    - Notify Slack
- Even a two-step pipeline (push → deploy) is a valid and useful CI/CD setup.



LET'S GET ON THE SAME PAGE

What is Continuous Development/Delivery/Deployment?

- The act of putting your code into an environment where it runs, such as production or staging.
- Deployment is the act of handing your code the keys to the house and saying "go live."
- Ways to deploy:
  - SFTP or Rsync
  - Docker container push
  - CI/CD pipeline-triggered deployment
  - You *can* use FTP, but it's strongly discouraged.

LET'S GET ON THE SAME PAGE

What is Continuous Development/Delivery/Deployment?

- The act of putting your code into an environment where it runs, such as production or staging.
- Deployment is the act of handing your code the keys to the house and saying "go live."
- Ways to deploy:
  - SFTP or Rsync
  - Docker container push
  - CI/CD pipeline-triggered deployment
  - ~~You *can* use FTP, but it's strongly discouraged.~~

CI/CD TOOLS AND PIPELINES: AUTOMATING DEPLOYMENTS WITHOUT BREAKING THE BANK

REPOSITORY PLATFORMS

WHAT IS THE PURPOSE OF A REPOSITORY PLATFORM?

KEEP TRACK OF YOUR CODE SO YOU'RE NOT PLAYING "WHO OVERWROTE WHAT" AT 2 AM

REPOSITORY PLATFORMS

WHAT IS THE PURPOSE OF A REPOSITORY PLATFORM?

NO MORE LIVING IN FEAR OF THE JUNIOR DEVELOPER ACCIDENTALLY SAVING

FINAL_V9_REALLYFINAL.CFM

TO PRODUCTION.

REPOSITORY PLATFORMS

WHAT IS THE PURPOSE OF A REPOSITORY PLATFORM?

THESE PLATFORMS ARE THE LAUNCHPAD. ONCE YOU PUSH CODE, THE PIPELINES DO THE HEAVY LIFTING.

TL;DR

TL;DR

- **GitHub Actions (Workflows):**
  - Built into GitHub, tons of prebuilt “actions.”
  - Great balance of power and accessibility.
  - Feels like the friend who always knows the cool new tools.
- **GitLab Pipelines:**
  - Pipelines broken into stages → jobs → scripts.
  - Everything in one place: repo, issues, registry, pipelines.
  - Perfect if you want less tool-sprawl and more control.
- **Bitbucket Pipelines:**
  - YAML + Docker, simple and reliable.
  - Works best if you’re already using Jira/Confluence.
  - The practical, no-frills option that still gets you home on time.

- The 800-pound gorilla: everybody uses it, even your cousin’s cat project.
- Free private repos so you don’t have to beg your boss for $7/month.
- GitHub Actions: click a button and suddenly you’re “doing DevOps.”
- Marketplace full of templates. Some are brilliant, some… let’s just say are “questionable.”

- GitHub doesn’t call their CI/CD tools “pipelines”.
- In GitHub, pipelines are driven by **Workflows** powered by **Actions**.

- A workflow is the pipeline. It’s a collection of actions that represent the entire pipeline flow.
- An action is like a Lego brick that does one specific job (like “run tests” or “deploy to AWS”).

1. The **Trigger**: Workflows kick off when something happens in your repo:
   - When you push to a main branch
   - Opening a pull request
   - On a schedule (like a nightly build)
   - Or even a manual “hit the button” deploy.
2. **Jobs**: A *workflow* is made of *jobs* that run on GitHub’s cloud runners (or your own servers if you’re fancy).
   - Each job runs in a clean environment (Linux, Windows, or Mac).
   - Jobs can run in parallel (faster) or depend on each other (safer).
3. **Steps**: Jobs contain *steps*, and steps run *actions*.
   - Example: actions/checkout@v4 (check out your code)
   - Example: actions/setup-node@v3 (install Node.js)
   - You can mix prebuilt actions with your own scripts.
4. **Artifacts & Results**: Workflows can:
   - Build and test your app.
   - Save build artifacts (like compiled code or Docker images).
   - Deploy to staging/production automatically when conditions are met.

Why It’s Useful for Small Shops

- Free minutes for public repos, generous quota for private ones.
- Tons of prebuilt actions so you don’t reinvent the wheel.
- Integrated with GitHub so there are no extra logins, no bolted-on tools.
- Great way to go from “cowboy coding” to an *actual process* without spending a dime.

Area	What's Included / Free Tier	What Costs Extra / Overage
Plans & User Fees	**Free plan:** $0 per month. Unlimited public and private repos. **Team plan:** ~$4 per user per month. Adds more features like repository rules, more Actions minutes, etc. **Enterprise Cloud plan:** ~$21 per user per month. For large orgs, enhanced security, compliance, etc.	More users means more licensing costs. Add-ons like Advanced Security, Codespaces, etc., bring extra charges.

Area	What's Included / Free Tier	What Costs Extra / Overage
GitHub Actions (CI/CD minutes + artifacts storage)	Included quotas depend on your plan: **Free Plan:** ~2,000 minutes and 500 MB storage/month. **Team Plan:** ~3,000 minutes and 2 GB storage per month. **Enterprise Cloud Plan:** ~50,000 minutes and 50 GB storage.	Using more minutes than your plan's quota means you pay per minute. Rates depend on runner type (OS, cores). For example, Linux runners are cheapest, Windows costs more, and macOS costs much more. Large runners and more powerful machines cost more. Charges apply for Extra artifact and package storage beyond free allotment.

Area	What's Included / Free Tier	What Costs Extra / Overage
GitHub Packages and Storage / Bandwidth	Some free storage & bandwidth for packages depending on the plan. Public repos often get generous or free usage.	If you exceed the free storage or bandwidth, you pay overage fees. Also, egress (data transfer out) may cost in some tiers.
Codespaces, Codespaces Storage and Compute	Free or included hours / storage in certain plans (lower for Free plans, more for paid plans).	Using more compute hours than included, larger dev environments, or utilizing more storage will incur extra fees.

- GitLab doesn't try to rebrand pipelines as "workflows" or "actions." They just call them **Pipelines**.
- Defined in `.gitlab-ci.yml`. A single YAML file that looks innocent until you realize you'll spend half your day debugging indentation.
- The hierarchy: **Pipeline** → **Stages** → **Jobs**. Think of it like Russian nesting dolls, but with more failed builds.

1. **Trigger:** Pipelines kick off when you push, merge, or click the “Run Pipeline” button. Basically, GitLab is your mom yelling, “Clean up your mess!” every time you commit.
2. **Stages:** Pipelines are split into stages (e.g., build, test, deploy).
   - Stages run in order, like soldiers in line.
   - Inside a stage, jobs can run in parallel - because why not burn all your runner minutes at once?
3. **Jobs:** Each job = the actual grunt work.
   - Runs scripts you define.
   - Executes on runners (GitLab’s minions).
   - Use their shared runners, or host your own if you love server maintenance as a hobby.
4. **Artifacts & Results:**
   - Store artifacts (builds, reports, packages) so you can prove you *actually* did work.
   - Pass them between stages like hot potatoes.
   - Auto-deploy when all the green checkmarks line up like a DevOps slot machine.

Why It’s Useful for Small Shops

- **All-in-one:** Repos, issues, pipelines, and registry. There is less tool sprawl, and more single point of failure.
- **Free tier:** Unlimited private repos, plenty of pipeline minutes… until you burn through them by accident.
- **Self-hosting:** You can run GitLab on your own hardware if you enjoy pain.
- **Docker + Kubernetes integration:** Because nothing says “weekend ruined” like Helm charts.

Area	What's Included / Free Tier	What Costs Extra / Overage
Plans & User Fees	**Free plan:** no user fee and includes basic CI/CD and repo features. **Premium plan:** ~$29 per user per month* **Ultimate plan:** ~$99 per user per month* * billed annually. I hate this.	More advanced security/compliance/ planning features in the Premium and Ultimate plans. Higher levels of support. More included compute (CI minutes) and storage.

Area	What's Included / Free Tier	What Costs Extra / Overage
CI/CD Compute Minutes (Shared Runners)	Included quotas depend on your plan: **Free Plan:** 400 minutes per month on the GitLab shared runners. **Premium Plan:** ~10,000 minutes per month. **Ultimate Plan:** ~50,000 minutes per month.	If you exceed the included minutes, you must either pay for extra shared runner minutes, or use your own runners, which don't count against the shared-quota, but increase your CI/CD management by a lot.

Area	What's Included / Free Tier	What Costs Extra / Overage
Storage (Repos and LFS)	**Free plan visibility:** ~10 GB per project (Git repo and LFS) for Free users. **Premium and Ultimate** have much larger storage per project (500 GB for many plans).	If you need more storage overall, beyond what's included, there may be extra cost or limits. Also note: large file storage, repository size limits per project may enforce caps.
Other Features and Add-Ons	Some security scanning, code quality, merge request approvals etc., in paid plans. Free users get basic CI/CD, issue tracking etc.	Ultimate tier adds more compliance, vulnerability scanning, governance dashboards, etc. Also better support SLAs.

- Atlassian went with the obvious: **Pipelines**.
- Config lives in `bitbucket-pipelines.yml`. It’s simple, predictable, not going to win any beauty contests, but it works.
- If you’ve seen GitLab pipelines, this feels like the slimmed-down, more streamlined cousin.

1. **Trigger:** Kicks off on a push. Easy.
2. **Steps:** Pipelines are a series of steps run in Docker containers.
3. **Parallelism:** Run steps one after another, or spread them out to get results faster.
4. **Deployment Environments:** Define staging, test, and production right in the file so your team looks extra professional.

Why It’s Useful for Small Shops

- **Jira integration:** If your tickets already live in Jira, Bitbucket keeps everything in the family.
- **Free private repos:** Good fit for small teams that don’t want to worry about costs.
- **Docker-based execution:** If it works in a container, it works here.
- **Doesn’t overwhelm you with features.** BitBucket is just enough to get a solid pipeline running without too many distractions.

Area	What's Included / Free Tier	What Costs Extra / Overage
Plans & User Fees	**Free Plan:** Good start for very small teams. Unlimited private repos with basic features. **Standard Plan:** ~$3 per user per month. More CI/CD minutes, and more features like improved merge checks, etc. **Premium Plan:** ~$6 per user per month. Adds security and administrator controls such as deployment permissions, IP whitelisting, etc.	If you exceed included build minutes, you buy extra minutes in "build packs" (blocks of minutes). More expensive features and tighter security or enforcement controls are in higher plans. More users = higher costs.

Area	What's Included / Free Tier	What Costs Extra / Overage
CI/CD / Build Minutes	Included quotas depend on your plan: **Free Plan:** 50 minutes per month included for Pipelines. **Standard plan:** ~2,500 build minutes per month. **Premium plan:** ~3,500 build minutes per month.	If you go above your plan's included minutes, you buy extra minutes in packs of 1,000 minutes for $10 per month per pack.

Area	What's Included / Free Tier	What Costs Extra / Overage
Features & Security / Admin Controls	Basic code review, unlimited private repos, standard integrations, basic branch permissions etc. included in Free/Standard.	Premium adds more advanced controls: enforced merge checks, IP allow-listing, deployment permissions, stronger auditing etc.
How Usage Scales / Overage Costs	Many small teams use much less than their minute allowance in Standard or Premium; Free plan sometimes gets enough to do basic work.	Extra build minutes are the main variable cost. Also, having many users, large repos, many test runs or long build steps will push you toward needing more minutes.

SUPER TL;DR

SUPER TL;DR

- **GitHub** → flashy, friendly, and packed with integrations.
- **GitLab** → powerful and all-inclusive; one platform to run the show.
- **Bitbucket** → straightforward and dependable, especially for Atlassian users. If you like simple, this is the way to go.

PLATFORM NATIVE REPOSITORIES

TL;DR

TL;DR

- **Amazon Web Services**
  - Fully managed Git repos on AWS.
  - Plays nicely with CodePipeline, IAM, and the rest of the AWS zoo.
  - Solid choice if your whole stack already lives in AWS.
- **Google Cloud Platform**
  - Git repos tucked neatly into Google Cloud.
  - Integrates with Cloud Build, IAM, and GCP logging.
  - Great if you’re already drinking the Google Kool-Aid.
- **Microsoft Azure**
  - Part of Azure DevOps.
  - Strong enterprise features, branch policies, and approvals.
  - Works best if your world revolves around Microsoft.

- AWS doesn’t reinvent Git. It’s just **CodeCommit** repositories.
- Fully managed, private Git repos hosted in AWS.
- Think of it as “GitHub, but inside the AWS walled garden.”
- AWS CodeCommit is dirt-cheap if you’re already in AWS. First 5 users are free, and even bigger teams rarely pay more than a couple of lattes per month. Unless of course your builds are enormous or you’re hammering the repo constantly.

1. **Trigger:** Pushing to CodeCommit can trigger events in AWS **CodePipeline** or **CodeBuild**.
   - Example:
     - push to main →
     - pipeline kicks off build →
     - deploys to EC2, Lambda, or wherever you point it to.
2. **Integration:** Deeply tied into the AWS ecosystem (IAM for access, CloudWatch for logging, CodeDeploy for rollout).
3. **Runners:** Unlike the SaaS hosted repos, you don’t configure runners. You wire CodeCommit into other AWS services that do the building and deploying.
4. **Flow**: Commit → Pipeline runs → Build and Deploy → Results get tracked in CloudWatch.

Why It’s Useful for Small Shops

- **Seamless with AWS stack:** If you’re already hosting in AWS, it keeps everything under one roof.
- **Security baked in:** Uses AWS Identity Access Management (IAM), so you don’t need to manage another user system.
- **No repo size limits:** Unlike GitHub’s “let’s argue about 1 GB files,” CodeCommit is more forgiving.
- **Pay-as-you-go:** No per-seat licensing. You pay for usage, which can be cheaper for small teams.
- **Downside:** The learning curve is pure AWS. It’s powerful, but sometimes you feel like you need a PhD in IAM policies just to push code.

Area	What's Included / Free Tier	What Costs Extra / Overage
Plans & User Fees	The first 5 **active users** free. An "active user" is any IAM user/role/federated identity that accesses CodeCommit that month.	After that: $1.00 per active user per month.
Storage	Each active user includes **10 GB-months** storage.	Extra storage: **$0.06 per GB-month** beyond included.

Area	What's Included / Free Tier	What Costs Extra / Overage
Git Requests	Each active user includes **2,000 Git requests** per month. (push/pull, etc.).	Extra requests: **$0.001 per request** beyond the included allotment.
Repo Limits	Unlimited repos for free.	No repo-count charges; only storage and requests matter.
Other Services	Deep integration with AWS CodePipeline, CodeBuild, CodeDeploy, IAM, CloudWatch.	Using those services incurs **separate charges** (pipelines, builds, deploys, logging, etc.).

Example Scenarios

Tiny team (≤5 devs, light use):

- Cost: $0 per month if under 10 GB storage and under 2,000 requests per user.
- Great for hobby projects or small shops already on AWS.

Small team (10 devs, moderate use):

- 5 free active users, 5 billed for $5 per month.
- If they use ~20 GB total storage and ~3,000 requests per user, that’s ~$2–3 more in overages.

Growing team (20 devs, heavier use):

- 15 billed users are $15 per month.
- If they push/pull a lot (say 5,000 requests per user) and use 50 GB storage, expect ~$10–15/month extra.

Google Cloud Platform

What They’re Called

- Just plain old **Cloud Source Repositories** (no clever branding here).

- Fully managed Git repos that live inside your GCP project.
- Feels a bit like Google saying, “Oh yeah, we can do Git too.”

Google Cloud Platform

How They Work in CI/CD

1. **Trigger**: Pushing to a repo can kick off Cloud Build jobs (Google's CI/CD engine).
2. **Integration**: Works with IAM, Cloud Logging, Cloud Monitoring. Everything stays inside the Google walled garden.
3. **Builds**: Cloud Build runs jobs in containers on demand, with flexible machine types.
4. **Flow**: Push → Cloud Build builds/tests/deploys → Deploy to GCP services (App Engine, Cloud Run, GKE, etc.).

Google Cloud Platform

Why It’s Useful for Small Shops

- The free tier is generous: 50 GB storage, 50 GB outbound traffic, and 5 active users per project at no cost.
- 2,500 free Cloud Build minutes per month means a small team can often run entirely free.
- Tight integration with GCP services: Deploying to Cloud Run, App Engine, or GKE is almost push-button.
- Simple, lightweight option: No extra bells and whistles. GCP is just Git repos with solid hooks into Google’s infrastructure.

Google Cloud Platform

Area	What's Included / Free Tier	What Costs Extra / Overage
Cloud Source Repositories	Free storage up to 50 GB per month across all repositories. Free outbound network data transfer up to 50 GB per month. Up to 5 "project-users" free per project (i.e. people who access or modify repos) without user-charges.	More than 5 project-users in a project incurs an $1 per extra project-user per month. Storage beyond 50 GB incurs $0.10 per GB per month Network egress beyond 50 GB per month incurs $0.10 per GB per month.

Google Cloud Platform

Area	What's Included / Free Tier	What Costs Extra / Overage
Cloud Build (CI/CD builds, tests, etc.)	Free tier: 2,500 build-minutes per month (for certain machine types) under the default pool. Also free is 100 GB SSD disk space for build VM up to first 100 GB for default pools.	After free minutes, you pay per minute depending on machine type. Some sample rates: - e2-standard-2 (2 vCPU / 8 GB RAM) ~ $0.006 / minute. - Higher CPU or memory machines cost more (e.g. e2-highcpu-8, etc.) If using private pools or bigger specs (more vCPUs/memory), those can get significantly higher per-minute rates.

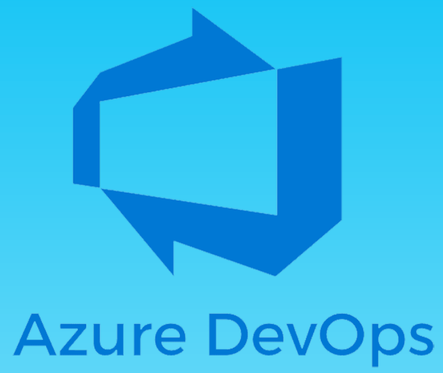
Google Cloud Platform

Example Scenarios & Implications for Small Shops

- If your team has ≤5 people, and you keep repo size modest (under 50 GB), and don't exceed 50 GB egress, you'll likely pay $0 for Source Repos.
- If builds and tests are small and light and you're okay with basic machine types, you can stay within the free 2,500 build-minutes per month in Cloud Build for a while.
- Once you need bigger machines, more users, or a lot of data transfer, costs start scaling. But relative to many paid plans of other platforms, GCP can be pretty reasonable; especially if you already host other services there.

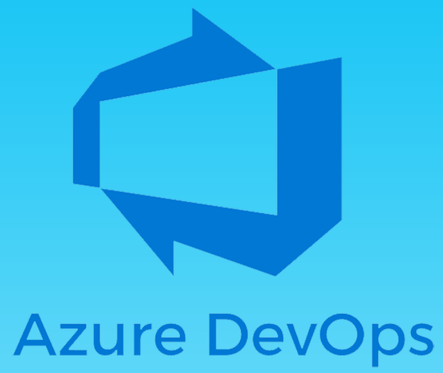
What They’re Called

- Simply Azure Repos. It's part of the broader Azure DevOps suite.
- Git-based repositories with unlimited private repos.
- Comes bundled with other Azure DevOps tools (Boards, Pipelines, Artifacts, Test Plans, etc.).



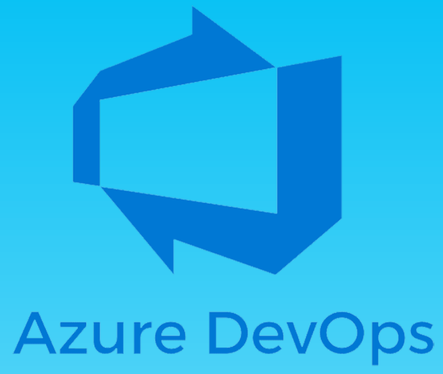
How They Work in CI/CD

1. **Trigger:** Commits or pull requests in Azure Repos trigger Azure Pipelines (Microsoft's CI/CD engine).
2. **Integration:** Deep ties into Azure AD for identity, plus the rest of Azure services.
3. **Pipelines:** YAML-based or visual designer pipelines, supporting multi-stage builds and deployments.
4. **Flow:** Push → Pipeline runs → Build/test → Deploy to Azure services (VMs, App Service, AKS, etc.).

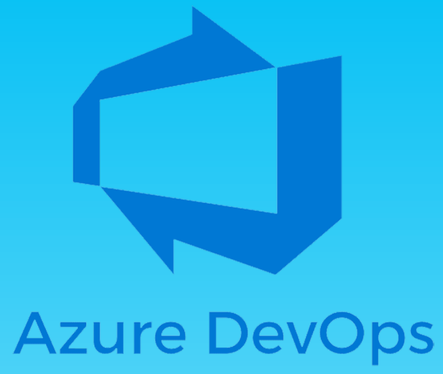


Why It’s Useful for Small Shops

- Unlimited private repos even on the free tier.
- 5 free users with Azure DevOps; beyond that it’s per-user per month.
- 2,000 free pipeline minutes per month on Microsoft-hosted agents, or unlimited if you self-host agents.
- Great if you’re already in the Microsoft ecosystem (Azure hosting, Office 365, AD integration).
- Strong enterprise-grade features (branch policies, gated check-ins) without extra setup.



Area	What's Included / Free Tier	What Costs Extra / Overage
Plans & User Fees	Azure DevOps includes 5 free Basic users per organization, plus unlimited stakeholders at no cost. Stakeholders are read-only and work tracking.	Beyond 5 users: $6 per user per month for each extra Basic user. Advanced features (Test Plans, advanced reporting) are add-ons at higher cost.
Repositories	Unlimited private Git repos included in all plans.	No repo-related overages.



Area	What's Included / Free Tier	What Costs Extra / Overage
CI/CD (Azure Pipelines minutes)	2,000 free minutes per month on Microsoft-hosted agents. 1 free self-hosted agent with unlimited minutes	Extra Microsoft-hosted agent minutes are $40 per parallel job per month (with 1,800 minutes included). Additional self-hosted parallel jobs are $15 per job per month with unlimited minutes per job.
Storage (Artifacts, etc.)	The first 2 GB of Azure Artifacts storage per organization are free.	Beyond that it's $2 per GB per month for extra storage.
Other Features	Integration with Boards, Repos, Pipelines, Artifacts, and Test Plans in one suite.	Test Plans are a paid add-on (~$52 per user per month).

SUPER TL;DR

SUPER TL;DR

- **AWS CodeCommit** → reliable, no-frills, ideal if you're already knee-deep in AWS.
- **Google Cloud Source Repos** → streamlined, less flashy, but fits seamlessly into GCP pipelines.
- **Azure Repos** → enterprise-ready, feature-rich, and a natural pick for Microsoft shops.

BY THE WAY...

BY THE WAY...

- Deployment pipelines aren't limited to the big cloud providers.
- Many of the CF Summit sponsors can also assist in setting up a deployment pipeline to your hosted infrastructure.

ENVIRONMENTS:
STOP 👏 DEVELOPING 👏 ON 👏 PRODUCTION

ENVIRONMENTS

STOP 👏 DEVELOPING 👏 ON 👏 PRODUCTION

PLANES AREN'T REPAIRED IN MID-AIR.

ENVIRONMENTS

STOP 👏 DEVELOPING 👏 ON 👏 PRODUCTION

PROD IS *NOT* YOUR TEST LAB.

ENVIRONMENTS

STOP 👏 DEVELOPING 👏 ON 👏 PRODUCTION

GOAL: SAFE, BORING, REPEATABLE RELEASES.

THE MINIMUM SETUP (1-2 DEVS)

THE MINIMUM SETUP (1-2 DEVS)

1 or 2 Developers
Build and Test Locally

Push
code
to Repo

Triggers
Deployment
Pipeline

Deployed in Prod

THE MINIMUM SETUP (1-2 DEVS)

- Should mirror production as close as possible.
- Same web server, same database server.
- Same ColdFusion server, including the version.
- Check CFMLREPO.com for older versions.

THE MINIMUM SETUP (1-2 DEVS)

- Install ColdFusion Developer Edition Locally
- Run Ortus Solutions CommandBox
- Use the Adobe ColdFusion Docker images

THE MINIMUM SETUP (1-2 DEVS)

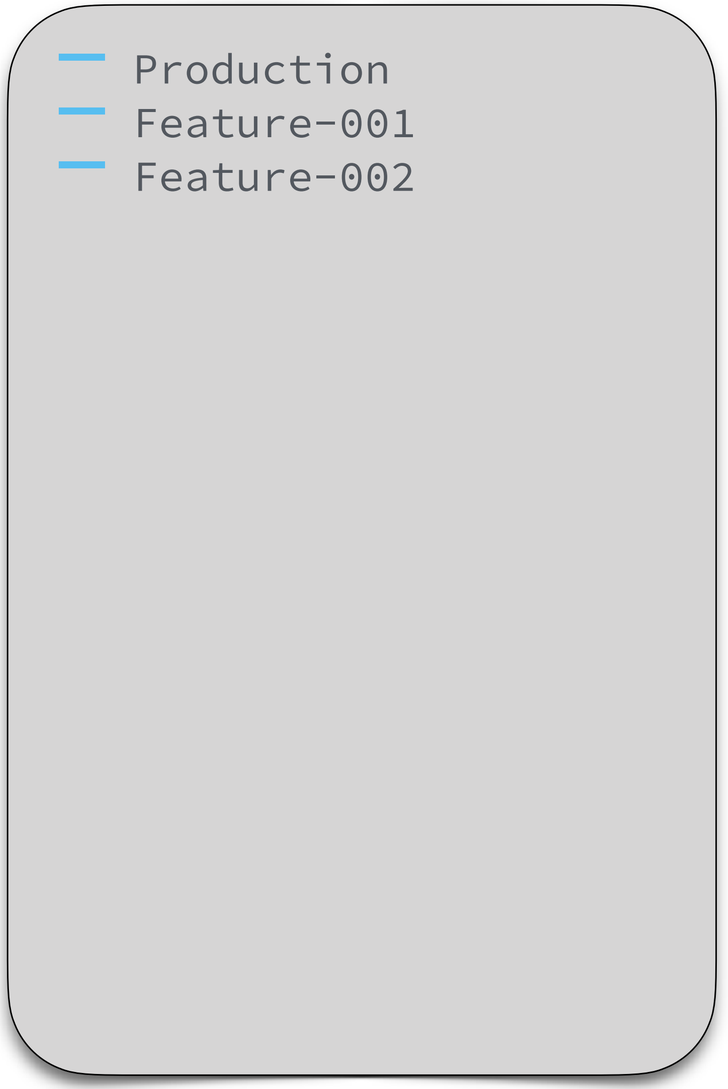
- Install a database server locally.
- Maybe the database is run remotely or over VPN.
- Use a copy of production to develop against.

DEPLOYMENT STRATEGIES: HOW TO SHIP LIKE A PRO

1-2 DEVELOPERS

- Small Application
- Developers build and test locally.
- Features merged to prod trigger deployment.

REPOSITORY BRANCHES



1-2 DEVELOPERS

- Small Application
- Developers build and test locally.
- Features merged to prod trigger deployment.

REPOSITORY BRANCHES

- Production
- Feature-002
- Feature-003
- Feature-001-Hotfix-001

- Single server
- Hosts website and database
- Single point of failure

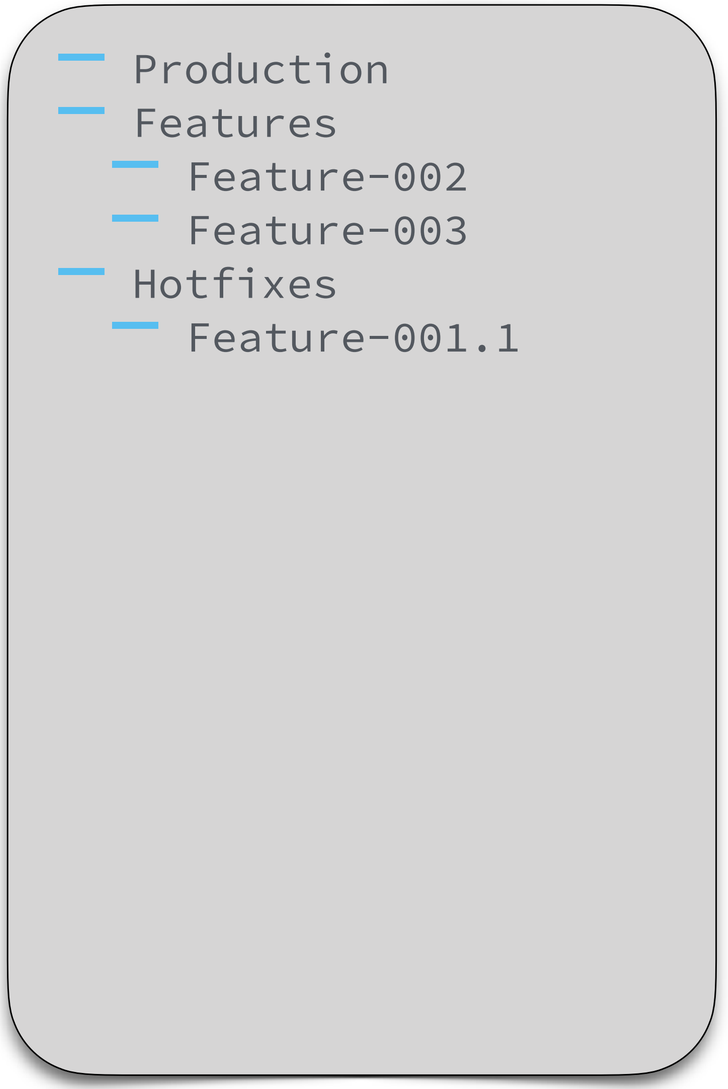
DATABASE SERVER

- Separate Web server and Database server
- More secure, but also more cost.
- Single point of failure for each

1-2 DEVELOPERS

- Small Application
- Developers build and test locally.
- Features merged to prod trigger deployment.

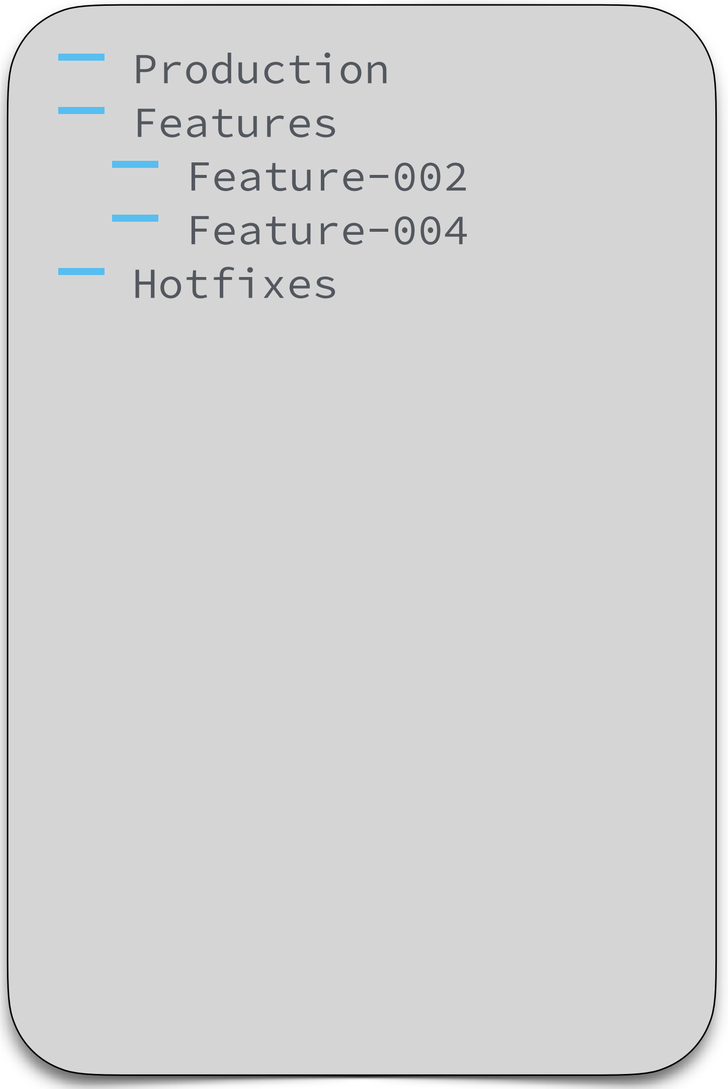
REPOSITORY BRANCHES



1-2 DEVELOPERS

- Small Application
- Developers build and test locally.
- Features merged to prod trigger deployment.

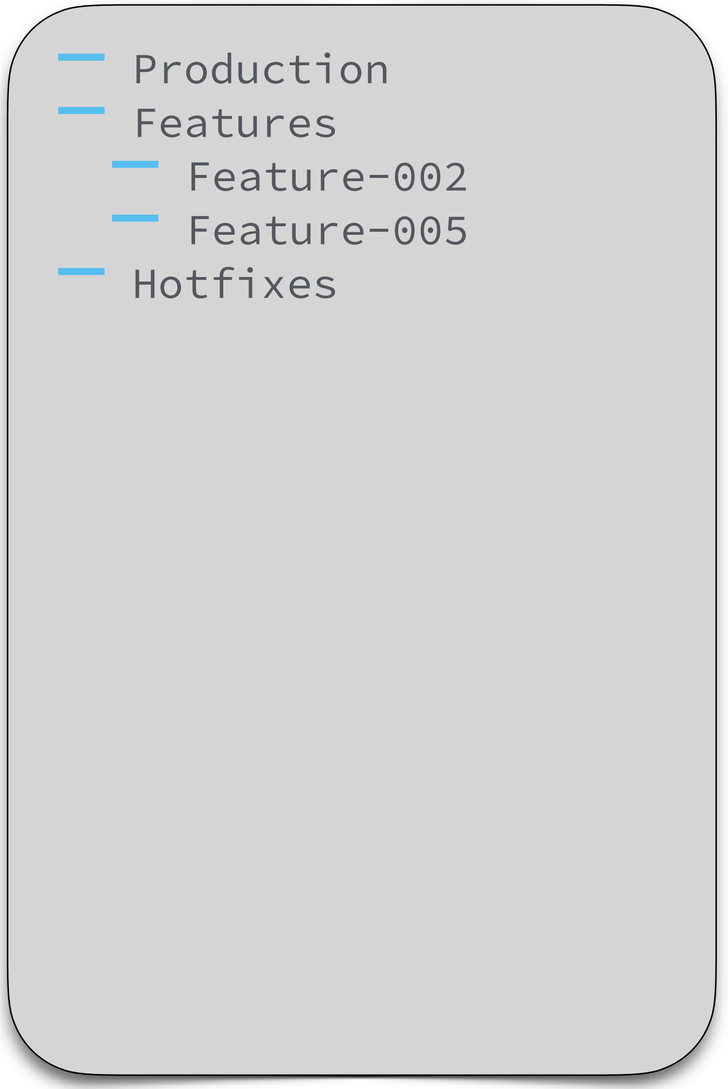
REPOSITORY BRANCHES



1-2 DEVELOPERS

- Small Application
- Developers build and test locally.
- Features merged to prod trigger deployment.

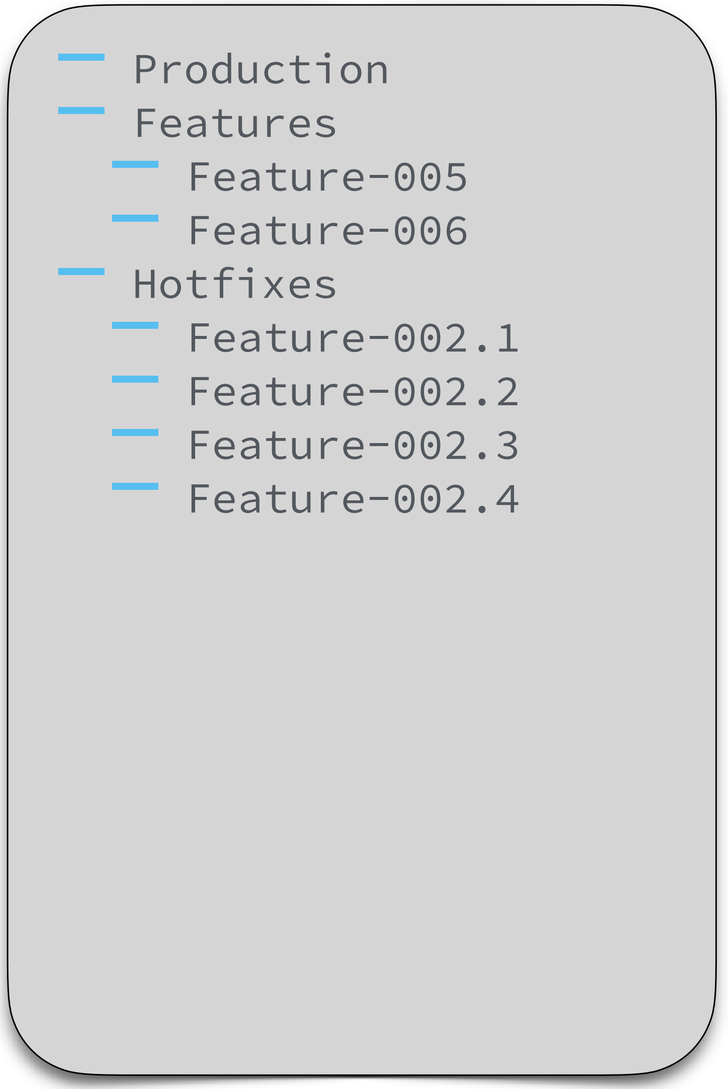
REPOSITORY BRANCHES



1-2 DEVELOPERS

- Growing Application
- Developers build and test locally.
- Features merged to prod trigger deployment.

REPOSITORY BRANCHES



3 DEVELOPERS

- Growing Application
- Developers build and test locally.
- Features merged to prod trigger deployment.

REPOSITORY BRANCHES

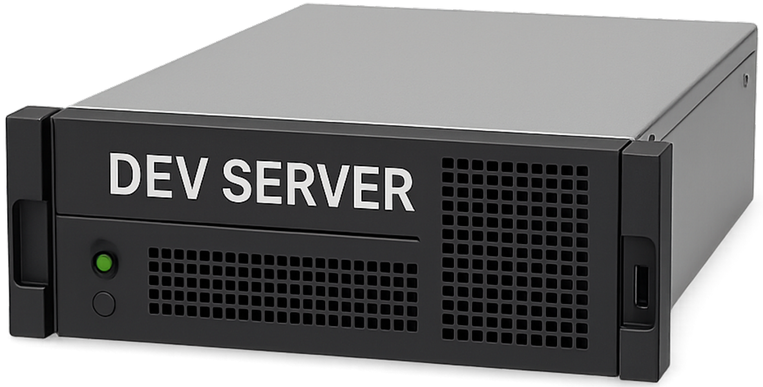
- Production
- Staging
- Features
  - Feature-005
  - Feature-006
- Hotfixes
  - Feature-002.1
  - Feature-002.2
  - Feature-002.3
  - Feature-002.4

A QUICK NOTE ABOUT STAGING ENVIRONMENTS

A QUICK NOTE ABOUT STAGING ENVIRONMENTS

- Your staging environment *can* be on your production server. It doesn't *require* its own server.
- It *CAN*.
- Doesn't mean it *should*.
- Having your staging environment co-hosted on your production server introduces risk to the live platform.
- It should be its own server. Maybe not as powerful as your production server(s), but isolated and configured exactly the same.

DATABASE SERVER



- Separate Production web server and Database server.
- Separate Development server.

- More secure, but also more cost.
- Keeps dev from taking down prod.

3 DEVELOPERS

- Growing Application
- Developers build and test locally.
- Features merged to prod trigger deployment.

REPOSITORY BRANCHES

- Production
- Staging
- Features
  - Feature-005
  - Feature-006
- Hotfixes
  - Feature-002.1
  - Feature-002.2
  - Feature-002.3
  - Feature-002.4

3 DEVELOPERS

- Growing Application
- Developers build and test locally.
- Developers merge *only* to Staging to test.
- Features merged to prod trigger deployment.

REPOSITORY BRANCHES

- Production
- Staging
- Features
  - Feature-005
  - Feature-006
- Hotfixes
  - Feature-002.1
  - Feature-002.2
  - Feature-002.3
  - Feature-002.4

3 DEVELOPERS

- Growing Application
- Developers build and test locally.
- Developers merge *only* to Staging to test.
- *Only* Staging merges to prod and *only* after full regression testing.
- Staging: the most up to date version of production plus any features or hotfixes about to be launched.
- When staging is merged to prod, it triggers deployment.

REPOSITORY BRANCHES

- Production
- Staging
- Development
  - Features
    - Feature-005
    - Feature-006
  - Hotfixes
    - Feature-002.1
    - Feature-002.2
    - Feature-002.3
    - Feature-002.4

3 DEVELOPERS

- Growing Application
- Developers build and test locally.
- Developers merge *only* to Staging to test.
- *Only* Staging merges to prod and *only* after full regression testing.
- Staging: the most up to date version of production plus any features or hotfixes about to be launched.
- When staging is merged to prod, it triggers deployment.

REPOSITORY BRANCHES

- Production
- Staging
  - v1.0.0
- Development
  - Features
    - Feature-005
    - Feature-006
  - Hotfixes
    - Feature-002.1
    - Feature-002.2
    - Feature-002.3
    - Feature-002.4

3 DEVELOPERS

- Growing Application
- Developers build and test locally.
- Developers merge *only* to Staging to test.
- *Only* Staging merges to prod and *only* after full regression testing.
- Staging: the most up to date version of production plus any features or hotfixes about to be launched.
- When staging is merged to prod, it triggers deployment.

REPOSITORY BRANCHES

- Production
- Staging
  - v1.0.0
  - next_release
- Development
  - Features
    - Feature-005
    - Feature-006
  - Hotfixes
    - Feature-002.1
    - Feature-002.2
    - Feature-002.3
    - Feature-002.4

3 DEVELOPERS

- Growing Application
- Developers build and test locally.
- Developers merge *only* to Staging to test.
- *Only* Staging merges to prod and *only* after full regression testing.
- Staging: the most up to date version of production plus any features or hotfixes about to be launched.
- When staging is merged to prod, it triggers deployment.

REPOSITORY BRANCHES

3 DEVELOPERS

- Growing Application
- Developers build and test locally.
- Developers merge *only* to Staging to test.
- *Only* Staging merges to prod and *only* after full regression testing.
- Staging: the most up to date version of production plus any features or hotfixes about to be launched.
- When staging is merged to prod, it triggers deployment.

REPOSITORY BRANCHES

- Production
- Staging
  - v1.0.0
  - v1.0.1
  - next_release
- Development
  - Features
    - Feature-005
    - Feature-006
  - Hotfixes

3+ DEVELOPERS

- Successful Application
- Developers build and test locally.
- Developers merge *only* to Staging to test.
- *Only* Staging merges to prod and *only* after full regression testing.
- Staging: the most up to date version of production plus any features or hotfixes about to be launched.
- When staging is merged to prod, it triggers deployment.

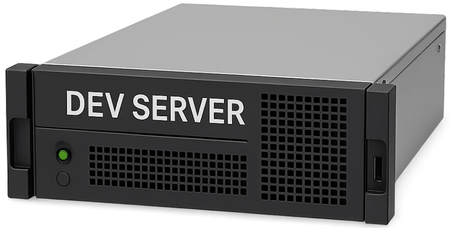
REPOSITORY BRANCHES

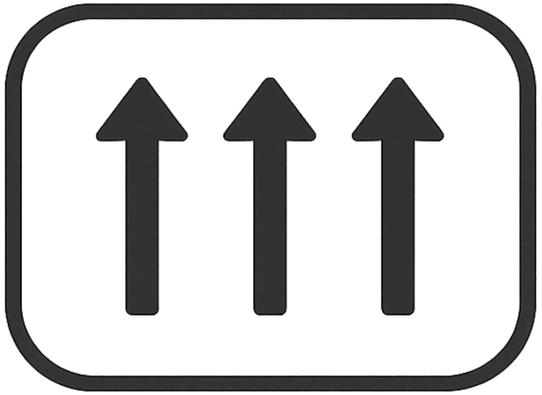
- Production
- Staging
  - v1.0.0
  - v1.0.1
  - v1.1.1
- Development
  - Features
  - Hotfixes

NOW YOU'RE SHIPPING LIKE A PRO!

DATABASE SERVER

DATABASE SERVER

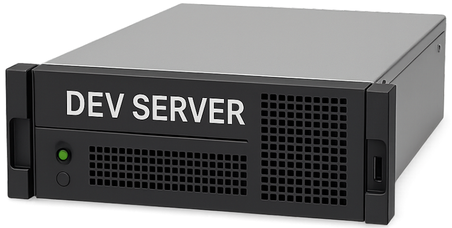




AUTO-SCALING GROUP

DATABASE SERVER

DATABASE SERVER

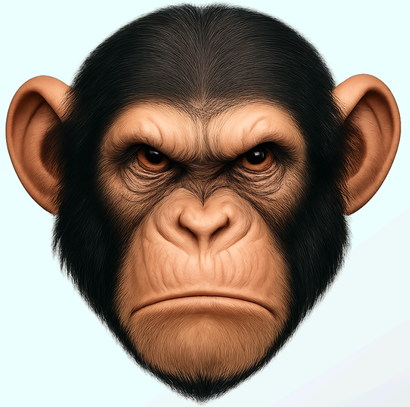


DO'S AND DON'TS

DO

- Use a pipeline to push code to servers.
- Use different databases for each environment.
- Sanitize your snapshots.

DON'T



- SSH into servers to alter code or FTP files.
- Use one database for everything.
- Use prod data in dev or local.

OTHER THINGS TO CONSIDER

THE ENVIRONMENT CONTRACT

- These are things to consider, if you're feeling fancy.
  - CFConfig for Admin settings.
  - Consider the 12-factor App Methodology (12factor.net): configs as env vars.
  - Try using feature flags for safer rollouts.
  - Docker containers are cattle, not pets.
  - `cfconfig export to=./config/staging.json`

SAFE DATA ACROSS ENVIRONMENTS

- **Golden Prod Snapshot**
  - Should be encrypted
  - Available on the VPC/VPN only.
- **Staging**
  - Data should be derived from the production snapshot
  - Personably Identifiable Information (PII) needs to be masked.
- **Local Seed**
  - small fixtures in Git.

CODE PROMOTION FLOW

- Passes local Tests → becomes a pull request.
- Another developer approves the pull request → push to staging.
- QA or UAT passes tests against staging → push to the release candidate.
- QA or UAT passes tests against release → code gets launched.

COLDFUSION FRIENDLY PRACTICES

- Dockerize your ACF environments using `adobe/coldfusion:[your CF version]`.
- Use CFConfig to synchronize all of your administrator settings across environments.
- Use the Logs.
- Application.cfc → read from environment vars.

COLDFUSION FRIENDLY PRACTICES

```
this.datasources["myDSN"] = {
    class: "com.microsoft.sqlserver.jdbc.SQLServerDriver",
    connectionString: "jdbc:sqlserver://" &
        server & ";databaseName=" & db & ";",
    username: server.system.environment["DB_USER"],
    password: server.system.environment["DB_PASS"]
};
```

COLDFUSION DOCKER COMPOSE EXAMPLE

```
services:
  coldfusion:
    image: adobe/coldfusion:2023
    ports:
      - "8500:8500"
    environment:
      - DB_USER=cfuser
      - DB_PASS=cfpass
    volumes:
      - ./app:/app
  db:
    image: mcr.microsoft.com/mssql/server:2022-latest
    environment:
      - ACCEPT_EULA=Y
      - SA_PASSWORD=SuperSecretPass1
    ports:
      - "1433:1433"
```

CFSUMMIT.VEGAS

Q&A

THANK YOU. IT TRULY IS A PRIVILEGE TO PRESENT FOR THE COLDFUSION COMMUNITY.

DAVID@CODEMONKEY.STUDIO