Explain the Fundamentals and Benefits of CI/CD to Achieve, Build, and Deploy Automation for Cloud-Based Software Products

Fundamentals of CI/CD:

Continuous Integration

*The practice of merging all developers' working copies to a shared mainline several times a day. It's the process of "**Making**". Everything related to the code fits here, and it all culminates in the ultimate goal of CI: a high quality, deployable artifact! Some common CI-related phases might include:*

- Compile
- Unit Test
- Static Analysis
- Dependency vulnerability testing
- Store artifact

Continuous Deployment

*A software engineering approach in which the value is delivered frequently through automated deployments. Everything related to deploying the artifact fits here. It's the process of "**Moving**" the artifact from the shelf to the spotlight. Some common CD-related phases might include:*

- Creating infrastructure
- Provisioning servers
- Copying files
- Promoting to production
- Smoke Testing (aka Verify)
- Rollbacks

Benefits of CI/CD

Technical Language	Value	Translation to the business's values
Catch Compile Errors After Merge	Reduce Cost	Less developer time on issues from new developer code
Catch Unit Test Failures	Avoid Cost	Less bugs in production and less time in testing
Detect Security Vulnerabilities	Avoid Cost	Prevent embarrassing or costly security holes
Automate Infrastructure Creation	Avoid Cost	Less human error, Faster deployments
Automate Infrastructure Cleanup	Reduce Cost	Less infrastructure costs from unused resources
Faster and More Frequent Production Deployments	Increase Revenue	New value-generating features released more quickly
Deploy to Production Without Manual Checks	Increase Revenue	Less time to market
Automated Smoke Tests	Protect Revenue	Reduced downtime from a deploy-related crash or major bug

Technical Language	Value	Translation to the business's values
Automated Rollback Triggered by Job Failure	Protect Revenue	Quick undo to return production to working state

Best Practices for CI/CD:

Fail Fast

Set up your CI/CD pipeline to find and reveal failures as fast as possible. The faster you can bring your code failures to light, the faster you can fix them.

Measure Quality

Measure your code quality so that you can see the positive effects of your improvement work (or the negative effects of technical debt).

Only Road to Production

Once CI/CD is deploying to production on your behalf, it must be the only way to deploy. Any other person or process that meddles with production after CI/CD is running will inevitably cause CI/CD to become inconsistent and fail.

Maximum Automation

If it can be automated, automate it. This will only improve your process!

Config in Code

All configuration code must be in code and versioned alongside your production code. This includes the CI/CD configuration files!

Deployment Strategy

- Big-Bang**:** Replace A with B all at once.
- Blue Green: Two versions of production: Blue or previous version and Green or new version. Traffic can still be routed to blue while testing green. Switching to the new version is done by simply shifting traffic from blue to green.
- Canary: Aka Rolling Update, after deploying the new version, start routing traffic to new version little by little until all traffic is hitting the new production. Both versions coexist for a period of time.
- A/B Testing: Similar to Canary, but instead of routing traffic to new version to accomplish a full deployment, you are testing your new version with a subset of users for feedback. You might end up routing all traffic to the new version, but that's always the goal.