Monitoring Services

17-316/616 Fall 2025

AI Tools for Software Development

https://ai-developer-tools.github.io

Austin Henley and Andrew Begel

Today

- Git Hygiene
- Deploying Services
- Monitoring Services

- Project Checkin Order
  1. Team CBD
  2. Team Bug Farmers
  3. Team Hive
  4. Team Prompt Engineers
  5. Team Code Cruise
  6. Team r/placeai

Practice good Git hygiene

1. Whenever working on a new task, create a new *feature* branch. *Never* work directly on the main branch.
2. Only when your feature is committed to the feature branch and fully tested, create a pull request (PR) to push the changes to the main branch.
3. Create a GitHub action that runs your integration tests on PR creation.
4. If your integration tests pass, have someone else on your team approve the PR.
5. Create a GitHub action to deploy to AWS on PR approval.

Secrets and key management

- Secrets include API keys, tokens, and passwords
- Many breaches come from hard-coded secrets or misconfigured automations!!!
- Do **not** store secrets in code
- Do **not** commit .env or config files with secrets
- Use GitHub Secrets (or AWS Secrets Manager)
  - Secure value store that is only accessible by GitHub Actions
  - ${{ secrets.API_KEY }}
- Have separate keys for dev/test/prod
- Know how to rotate and revoke all your secrets

Activity: Automate Deployment

- Get with your team
- Create a GitHub Action that deploys your app to AWS after each commit has been successfully tested and PR approved.
  - Code is committed, all tests pass, PR is approved and merged
  - Use GitHub Secrets to store your AWS secrets
  - Deploy your frontend
  - Deploy your backend
  - Send a Discord/Slack message when done.

We Deployed Our Service!

- Now what?
- How do we tell what's happening to it?
  - Is it running?
  - How well is it running?
  - Is it behaving as expected?

Common Risks with Cloud Services

- Availability
  - Did the service go down?
- Authentication
  - Can users log in?
- Authorization
  - Do users have appropriate permissions?
- Data Privacy
  - Did user data get leaked?
- Security
  - Did a crypto key get leaked or expire?
  - Is there some kind of cross-site scripting or DDOS attack occurring?

Additional Risks

- Database
  - Did a malicious user input inject code into your database?
- Deployment
  - Was a debug version of the app deployed instead of the production version?
- Migration
  - Was any user data lost when migrated from one database format to the next?

Logging

- Coding on your laptop
  - Emit strings from your code into a file on disk.
  - Explains what your code is doing
  - printf(“got to here.”)

Structured Logging

- How structured should our logs be?
  - All log output should record the timestamp.

  - printf(“%d, did stuff”, get_time())
  - INSERT INTO Log (time, title) VALUES (get_time(), ‘did stuff’)

Log Levels (from Node.js)

- Uncategorized messages
  - console.log(”my log message”)
- Log levels
  - Info: console.info(“my info message”)
  - Debug: console.debug(“my debug message”)
  - Warn: console.warn(“my warning”)
  - Error: console.error(“my error”)

What else should we log?

- Timestamp
- Log Category
- …

Where are the logs?

- When you run node, you can tell it where you want the logs to go:
  - node ./index.js > ./stdout-only.txt
  - node ./index.js 2> ./stderr-only.txt
  - node ./index.js 2>&1 ./stdout-and-stderr.txt
- Where are these files? Local machine or server?

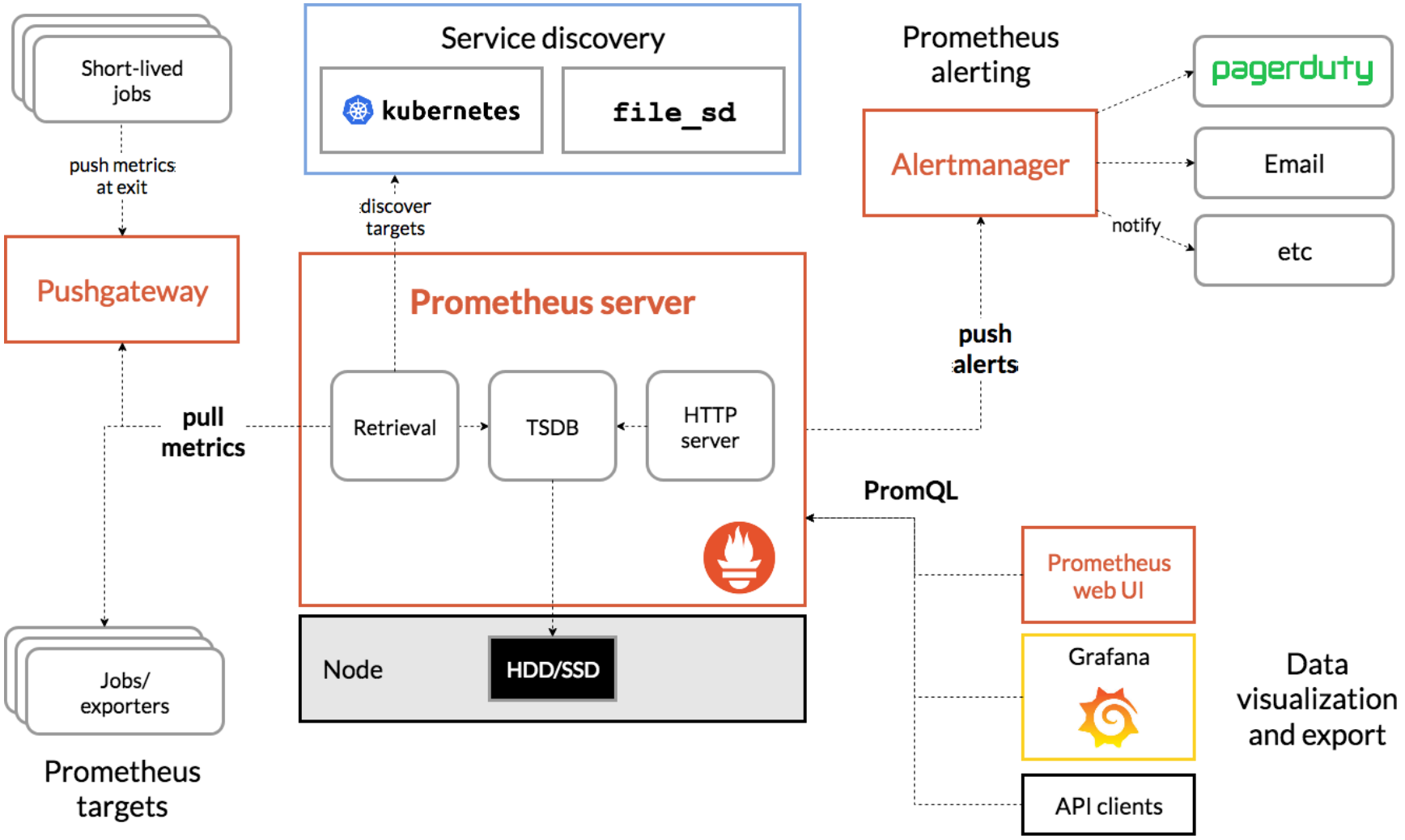
Node.js Log

```
[2025-02-19 13:04:00] INFO: Server started on port 3000
[2025-02-19 13:04:05] DEBUG: Received request: GET /
[2025-02-19 13:04:05] INFO: Serving index.html
[2025-02-19 13:04:10] WARN: User 'testuser' failed login attempt
[2025-02-19 13:04:15] ERROR: Database connection timeout
```

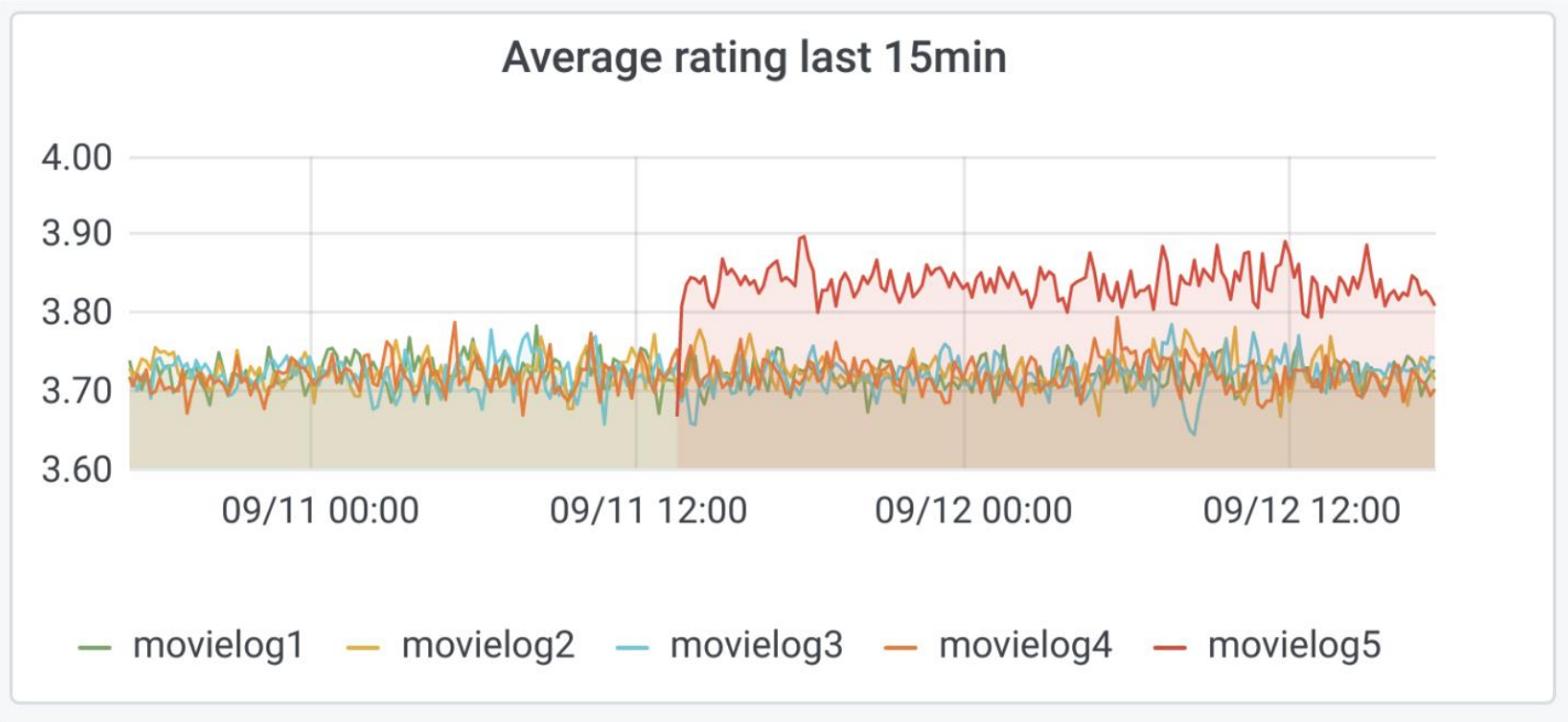
Monitor Your Logs with Dashboards

- Tools like Prometheus.io can attach to your logs and collect them for analysis.
- Grafana can visualize your logs with dashboards and charts.

Prometheus


Short-lived jobs
push metrics at exit
Pushgateway
Service discovery
kubernetes
file_sd
discover targets
Prometheus server
Retrieval
TSDB
HTTP server
pull metrics
Node
HDD/SSD
Jobs/ exporters
Prometheus targets
Prometheus alerting
Alertmanager
pagerduty
Email
etc
notify
push alerts
PromQL
Prometheus web UI
Grafana
API clients
Data visualization and export

Grafana Dashboard



Activity: AWS CloudWatch + Grafana

1. We assume your project is deployed to Amplify and Lambda
2. Go to AWS Console, Lambda, and your function
   - Click monitor and confirm you see graphs
3. Go to AWS Console, Cloud Watch, and look for log groups
   - If you see them, you are good
4. Go to AWS Console, Managed Grafana, and Create workspace
   - Open workspace and go to AWS data sources and Enable CloudWatch
   - Click Attach policy
5. Click Open Grafana workspace URL
6. Go to Dashboards, New, New dashboard, and Add new panel
   - Data source: CloudWatch
   - Namespace: AWS/Lambda
   - Metrics: invocations, errors, or duration

Next time

- No class Wednesday! Happy Thanksgiving!
- P2 Redo grades today.
- P7 released today.

- Final week!
  - Project Presentations
  - Course Wrapup – Lessons Learned
  - Course Improvement Feedback
  - Class party on Wednesday

- 2 teams will present Monday (volunteers?)
- 4 teams will present Wednesday