Continuous Deployment

17-316/616 Fall 2025

AI Tools for Software Development

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

Austin Henley and Andrew Begel

What should we be monitoring?

- We deployed, now what?
- How do we know it is working?
- What could fail?

- Let me tell you a story…

Administrivia

- Sign up for reflection if you haven’t
- Next Monday: Be ready for a quick demo and check-in about your project status

Let’s pick up where we left off…

Activity: Connect REST API to Calculator

- Modify the Lambda code so it is invoked by HTTP POST requests to your calculate() function.
- Step 1: Open the AWS API Gateway console.
  - Create API
  - In the REST API box, choose Build
  - Under API details, enter Calculator
- Step 2: In the Resources page for your API, choose Create Resource
  - ResourceName is CalculatorManager
- Step 3: Create an HTTP POST method
  - In Resources, highlight CalculatorManager. Choose Create Method
    - Method Type: POST
    - Integration type: Lambda
    - Lambda: enter LambdaFunctionOverHttps
- Step 4: Deploy the API
  - In Resources, choose Deploy API
    - Stage: New stage: test
  - Copy the invoke URL into your calculator frontend.
- This tutorial may help: https://docs.aws.amazon.com/lambda/latest/dg/services-apigateway-tutorial.html

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
  - BONUS: Send a Discord/Slack message

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?

Data to answer key questions

- Are our servers running as expected?
- Are our services working as expected?
- Who is accessing our data?
- How do our users behave?

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
- …

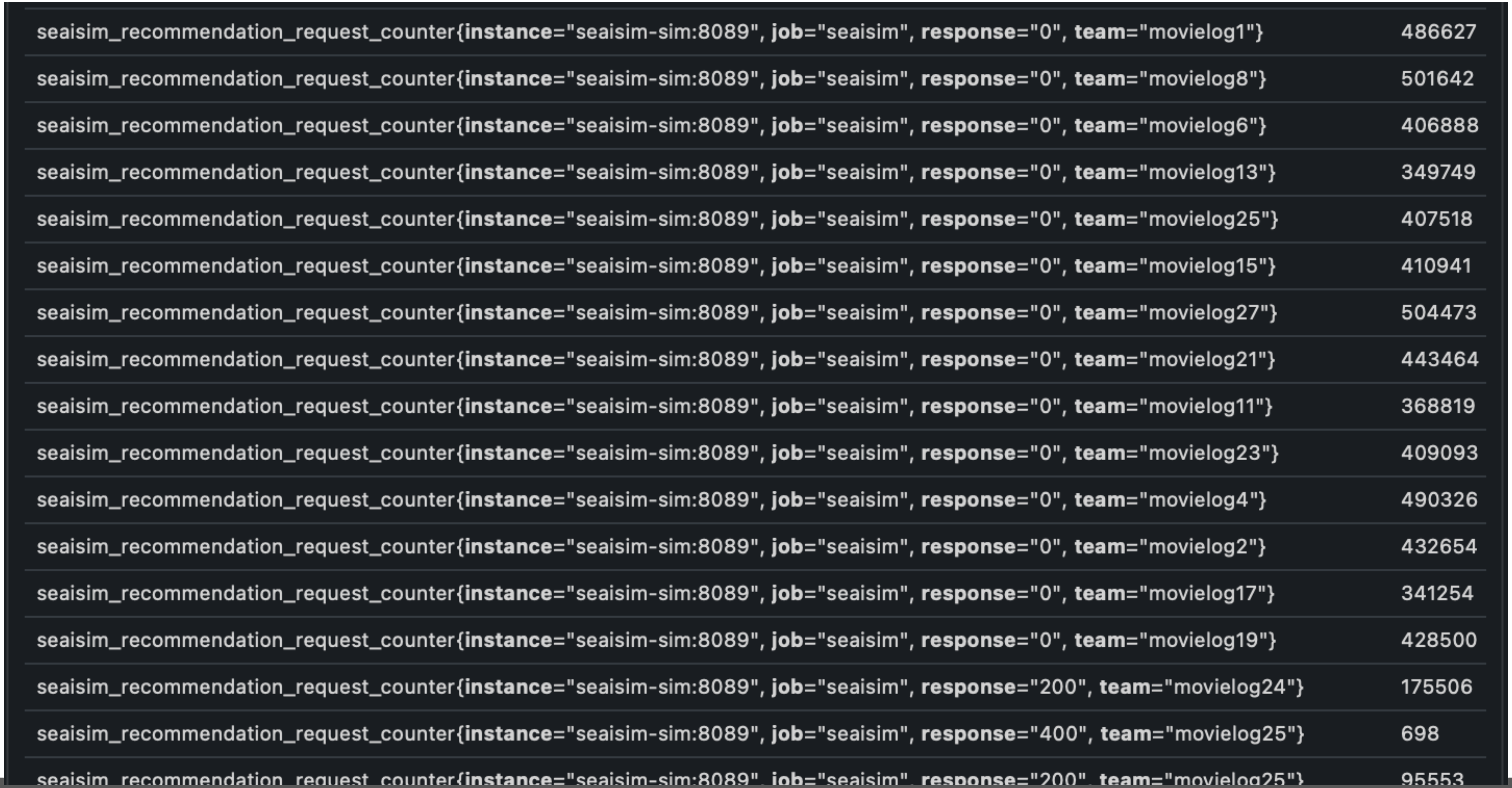
Where are the logs?

- 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
```

Prof. Henley’s App Logger



```
seaisim_recommendation_request_counter{instance="seaisim-sim:8089", job="seaisim", response="0", team="movielog1"}      486627
seaisim_recommendation_request_counter{instance="seaisim-sim:8089", job="seaisim", response="0", team="movielog8"}      501642
seaisim_recommendation_request_counter{instance="seaisim-sim:8089", job="seaisim", response="0", team="movielog6"}      406888
seaisim_recommendation_request_counter{instance="seaisim-sim:8089", job="seaisim", response="0", team="movielog13"}     349749
seaisim_recommendation_request_counter{instance="seaisim-sim:8089", job="seaisim", response="0", team="movielog25"}     407518
seaisim_recommendation_request_counter{instance="seaisim-sim:8089", job="seaisim", response="0", team="movielog15"}     410941
seaisim_recommendation_request_counter{instance="seaisim-sim:8089", job="seaisim", response="0", team="movielog27"}     504473
seaisim_recommendation_request_counter{instance="seaisim-sim:8089", job="seaisim", response="0", team="movielog21"}     443464
seaisim_recommendation_request_counter{instance="seaisim-sim:8089", job="seaisim", response="0", team="movielog11"}     368819
seaisim_recommendation_request_counter{instance="seaisim-sim:8089", job="seaisim", response="0", team="movielog23"}     409093
seaisim_recommendation_request_counter{instance="seaisim-sim:8089", job="seaisim", response="0", team="movielog4"}      490326
seaisim_recommendation_request_counter{instance="seaisim-sim:8089", job="seaisim", response="0", team="movielog2"}      432654
seaisim_recommendation_request_counter{instance="seaisim-sim:8089", job="seaisim", response="0", team="movielog17"}     341254
seaisim_recommendation_request_counter{instance="seaisim-sim:8089", job="seaisim", response="0", team="movielog19"}     428500
seaisim_recommendation_request_counter{instance="seaisim-sim:8089", job="seaisim", response="200", team="movielog24"}   175506
seaisim_recommendation_request_counter{instance="seaisim-sim:8089", job="seaisim", response="400", team="movielog25"}   698
seaisim_recommendation_request_counter{instance="seaisim-sim:8089", job="seaisim", response="200", team="movielog25"}   95553
```

Modern Logging Overview

- Collection and transport
  - All apps will write their own logs to disk
  - You need to collect these logs from where they're written and store them somewhere.
- Storage
  - Flat file or database
- Analysis
  - Grep/Findstr, SQL queries
- Alerts

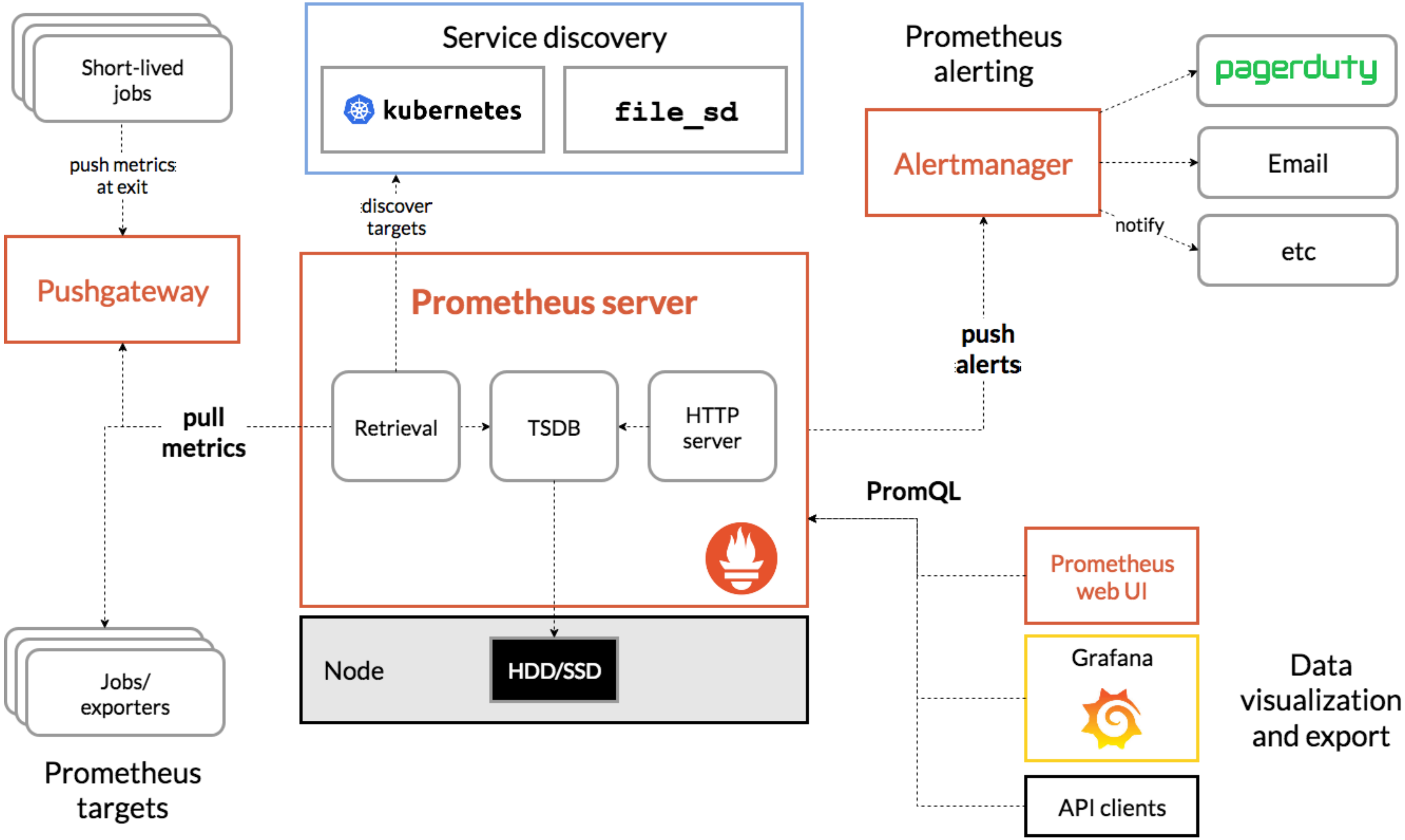
Collecting Logs across Computers

- All apps will write their own logs to disk
- You need to collect these logs from where they’re written and store them somewhere.
- Most logs are flat files
  - Easy to write
  - Easy to read
  - Can be interpreted later

Monitor Your Logs with Dashboards

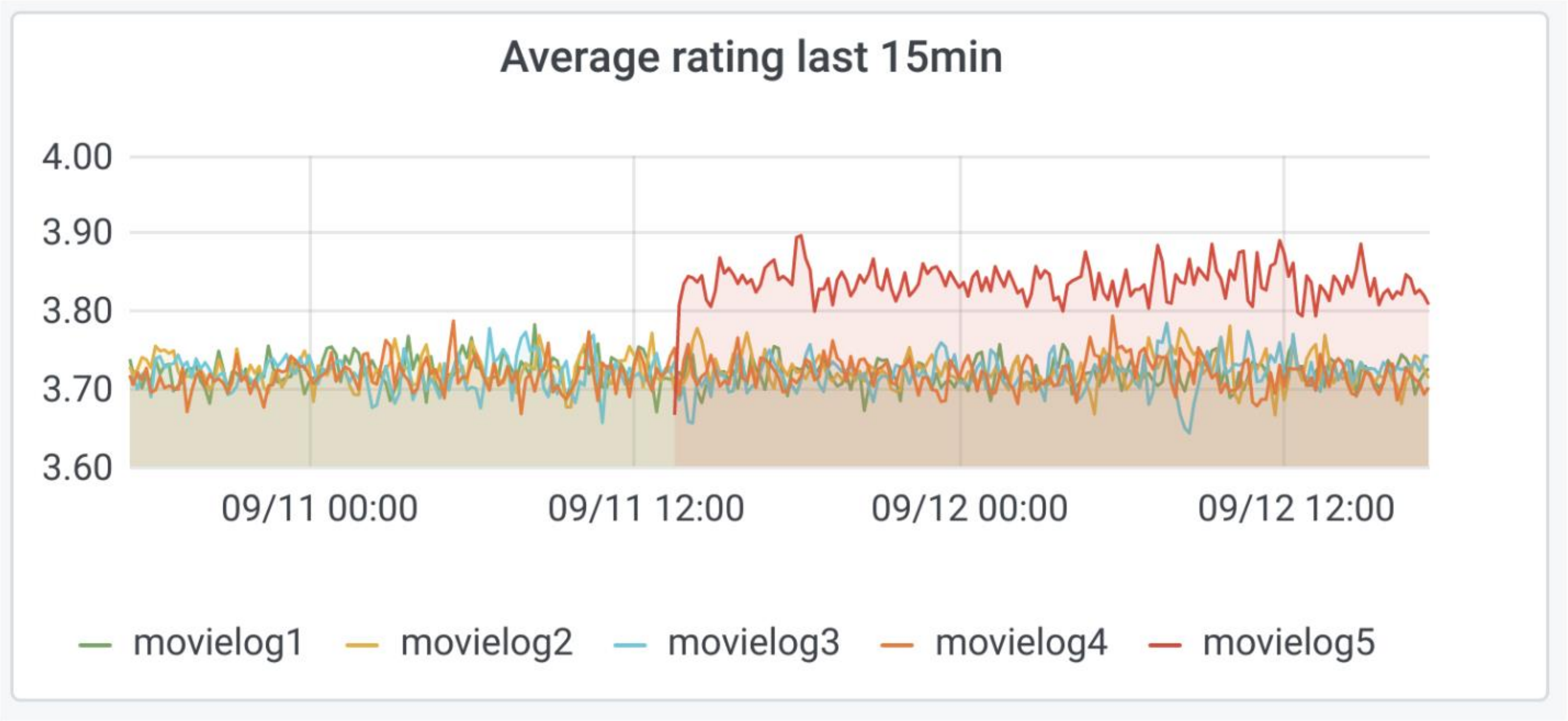
- Tools like Prometheus.io can attach to your logs and collect them for analysis.
- Grafana can visualize your logs with dashboards and charts.

Prometheus


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

Prometheus Demo

Grafana Dashboard



Dashboards

- https://abegel.grafana.net

Incident Management

- Developers live at HQ
- Operations live in the data center

- When operations is aware of a bug, they file a bug against the application.

- What kinds of cloud bugs are there?
- What is the impact to your Service Level Agreement (SLA)?

Problem Management Process

1. Problem management team accesses incident database
2. Impact analysis and prioritization
3. Root cause analysis
4. Escalate to appropriate development team
5. Fix the Problem

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?

Activity: Make calculator multi-user

Next class

- How do we know our deployment worked/is working?
- Cloud monitoring services
- E.g., Prometheus, Grafana, notifications