THE NTH REGION PROJECT

ANDREW BLOOMGARDEN
STAFF ENGINEER, NEW RELIC

Twitter: @aughr
micro.blog: @aughr
Mastodon: aughr@pgh.social
THE NTH REGION PROJECT

FORMAT

› What New Relic does
› What made this project difficult
› The technical changes we made
› The retro: the organizational challenges we faced
› Where we are now
New Relic Teams with IBM to Expand in Europe, Speed Cloud Adoption
RAILS MONOLITH

UI
API
DATA COLLECTION
Data growth at New Relic
2017
30

GIGABIT PER SECOND INBOUND
15,000,000

KAFKA MESSAGES PER SECOND
600 MILLION EVENTS PER MINUTE
~50
ENGINEERING TEAMS
HUNDREDS OF ENGINEERS
100%

DEV TEAMS ON CALL FOR THEIR SERVICES
EU region
Nth region
Aspiration: one small team can build a region
Project Backpack
MIGRATIONS AND DISASTER RECOVERY
MIGRATIONS

PROBLEMS TO SOLVE IN EIGHT YEARS

- Deploying many services
- Supporting a polyglot environment
- Service discovery
- Better secret management
- Container orchestration
MIGRATIONS

PROBLEMS TO SOLVE IN EIGHT YEARS

- Deploying many services
- Supporting a polyglot environment
- Service discovery
- Better secret management
- Container orchestration
- Multiple regions
MIGRATIONS ARE THE ONLY MECHANISM TO EFFECTIVELY MANAGE TECHNICAL DEBT AS YOUR COMPANY AND CODE GROWS. IF YOU DON'T GET EFFECTIVE AT SOFTWARE AND SYSTEM MIGRATIONS, YOU'LL END UP Languishing IN TECHNICAL DEBT.

Will Larson, April 2018
THE VINTAGES OF NEW RELIC

- 2010: Capistrano and Puppet
- 2013: Docker v0.x and Centurion
- 2013: Serveza (in-house service discovery)
- 2016: Vault
- 2017: Grand Central (in-house build/deploy) and Container Fabric (Mesos)
MIGRATIONS

CAPISTRANO AND PUPPET

Dr 2017 #3176

Closed  anonymous wants to merge 212 commits into master from dr-2017

Conversation  9  Commits  212  Files changed  81  +2,325 -286
desc 'Production environment'
task :production => :common do
  env_vars NEW_RELIC_JAVA_AGENT_ENVIRONMENT: 'production'
  set_current_environment(:production)

  env_vars AGENT_DB_USER: 'al_acs'
  env_vars AGENT_DB_PASSWORD: '…'

  host '….nr-ops.net'
  host '….nr-ops.net'
  host '….nr-ops.net'
end
desc 'Disaster Recovery environment'
task :recovery => :common do
  env_vars NEW_RELIC_JAVA_AGENT_ENVIRONMENT: 'recovery'
  set_current_environment(:recovery)

  env_vars AGENT_DB_USERNAME: 'agent_commands'
  env_vars AGENT_DB_PASSWORD: '…'

  host 'usw2v-dr-docker-8.dr.nr-ops.net'
  host 'usw2v-dr-docker-16.dr.nr-ops.net'
end
SERVICE DISCOVERY

handle = Serveza::Service.new('feature_flag', 1)
endpoint = handle.api
private final static Map<String, Map<Integer, String>> AGENT_DB_NAMES =
    ImmutableMap.<String, Map<Integer, String>>builder()
    .put(STAGING,
        ImmutableMap.<Integer, String>builder()
            .put(1,
                ImmutableMap.<Integer, String>builder()
                    .put(1, "localhost")
                    .put(2, "agentdb")
                    .build())
            .build())
    .put(LOCAL_STAGING,
        ImmutableMap.<Integer, String>builder()
            .put(1,
                ImmutableMap.<Integer, String>builder()
                    .put(1, "localhost")
                    .put(2, "agentdb")
                    .build())
            .build())
    .put(RECOVERY,
        ImmutableMap.<Integer, String>builder()
            .put(1, "usw2v-dr-agentdb-3.dr.nr-ops.net")
            .put(2, "usw2v-dr-agentdb-7.dr.nr-ops.net")
            .build())
    .put(PRODUCTION,
        ImmutableMap.<Integer, String>builder()
            .put(1,
                ImmutableMap.<Integer, String>builder()
                    .put(1, "localhost")
                    .put(2, "agentdb")
                    .build())
            .put(2,
                ImmutableMap.<Integer, String>builder()
                    .put(1, "localhost")
                    .put(2, "agentdb")
                    .build())
            .build())
    .put(DEVELOPMENT, ImmutableMap.of(1, "localhost")
    .put(INTEGRATION, ImmutableMap.of(1, "agentdb")
    .build());
SERVICE DISCOVERY

```java
.put(RECOVERY,
    ImmutableMap.<Integer, String>builder()
        .put(1, "usw2v-dr-agendb-3.dr.nr-ops.net")
        .put(2, "usw2v-dr-agendb-7.dr.nr-ops.net")
        .build())
```
**Migrations**

**Container Fabric and Grand Central**

```plaintext
- name: production
datacenter: chicago
instances: 5
cpus: 3
memory_mb: 1024
vips: [example.vip.example]
env_vars:
  NEWRELIC_LICENSE_KEY: "vault_secret_path:containers/teams/connectivity/production/portal-service/newrelic_license_key"
  NEW_RELIC_INSIGHTS_INSERT_API_ENDPOINT: "https://staging-insights-collector.newrelic.com/v1/accounts/1/events"
  NEW_RELIC_INSIGHTS_INSERT_API_KEY: "vault_secret_path:containers/teams/connectivity/production/portal-service/insights_
  NEW_RELIC_JAVA_AGENT_ENVIRONMENT: "production"
  FEATURE_FLAG_SERVICE_ENVIRONMENT: 'production'
  SERVEZA_BACKING_STORE_URI: 'http://staticserve.nr-ops.net/serveza/v1/production.yaml'
  FEATURE_FLAG_AWS_ACCESS_KEY: "example-access-key"
  FEATURE_FLAG_AWS_SECRET_KEY: "vault_secret_path:containers/teams/connectivity/production/portal-service/feature_flag_aws_region: us-east-1"
```

@aughr
- name: dr
  instances: 3
  cpus: 3
  memory_mb: 1024
  notify_on_stale_releases: false
  vips: [null]

env_vars:
  NEWRELIC_LICENSE_KEY: "vault_secret_path:containers/teams/connectivity/dr/portal-service/newrelic_license_key"
  NEW_RELIC_INSIGHTS_INSERT_API_ENDPOINT: "https://staging-insights-collector.newrelic.com/v1/accounts/1/events"
  NEW_RELIC_INSIGHTS_INSERT_API_KEY: "vault_secret_path:containers/teams/connectivity/dr/portal-service/insights_insert_a
  NEW_RELIC_JAVA_AGENT_ENVIRONMENT: "dr"
  FEATURE_FLAG_SERVICE_ENVIRONMENT: 'dr'
  FEATURE_FLAG_ALERTS_BASE_URI: 'http://alertservice-dr.dr.nr-ops.net/api'
  SERVERA_BACKING_STORE_URI: 'http://staticserve.dr.nr-ops.net/servera/v1/dr.yaml'
  FEATURE_FLAG_AWS_ACCESS_KEY: "null"
  FEATURE_FLAG_AWS_SECRET_KEY: "null"
  FEATURE_FLAG_AWS_REGION: "us-west-2"
INTERFACES
Interfaces separate the contract from the implementation
GETTING DATABASE CREDENTIALS

- File a ticket
- Wait for the DB team to add the credentials and share them
- Add them to your service configuration
- Deploy
GETTING DATABASE CREDENTIALS

- Programmatically declare your service needs access to the DB
- ...black box...
- Deploy
INTERFACES

GETTING DATABASE CREDENTIALS, REALITY

- Programmatically declare your service needs access to the DB
- DB team still manually adds credentials
- Deploy
GETTING DATABASE CREDENTIALS, REIMPLEMENTATION

- Programmatically declare your service needs access to the DB
- Credentials automatically generated once a human reviews the access request
- Deploy
Encapsulation lets small teams act independently
HIGH LEVERAGE INTERFACES
HIGH-LEVERAGE INTERFACES

SERVICE DISCOVERY: PROBLEMS

- Mix of:
  - Serveza (homegrown)
  - Hard-coded
  - Env vars
- Credential management unsolved
- No way to do static analysis
WHY STATIC ANALYSIS?

SERVICE
ALICE
DEPLOYING
HIGH-LEVERAGE INTERFACES

WHY STATIC ANALYSIS?

SERVICE
ALICE

BROKEN

SERVICE
BOB

NO SUCH SERVICE
WHY STATIC ANALYSIS?
WHY STATIC ANALYSIS?

SERVICE ALICE
HEALTHY

SERVICE BOB
HEALTHY
handle = Serveza::Service.new('feature_flag', 1)
endpoint = handle.api
SERVICE DISCOVERY

BOB_URL: please tell me where bob is
SERVICE DISCOVERY

BOB_URL: 'http://bob.local'

BOB_HOST: 'bob.local'

BOB_ENDPOINT: 'http://bob.local/path/to/api/i/use'
HIGH-LEVERAGE INTERFACES

SERVICE DISCOVERY

BOB_URL: 'discovery_path:bob'

BOB_URL: 'http://bob.local'
HIGH-LEVERAGE INTERFACES

SERVICE DISCOVERY

DATABASE_HOST: 'my-db.local'
DATABASE_PORT: '3306'
DATABASE_USERNAME: 'myuser'
DATABASE_PASSWORD: 'mypass'
DATABASE_NAME: 'my_schema'
HIGH-LEVERAGE INTERFACES

SERVICE DISCOVERY

DATABASE_URL:
'mysql://myuser:mypass@my-db.local:3306/my_schema'
HIGH-LEVERAGE INTERFACES

SERVICE DISCOVERY

DATABASE_URL: 'discovery_path:@mydb'
HIGH-LEVERAGE INTERFACES

SERVICE DISCOVERY AS DEPENDENCY INJECTION

- Services declare their dependencies
- Locations injected via env var in standard format (URLs)
- Credentials part of URLs
- Static analysis is possible
HIGH-LEVERAGE INTERFACES

CONTAINERS EVERYWHERE

- Interface between teams and machines
- Stateless in Container Fabric (Mesos)
- Stateful in containers controlled via Centurion or Ansible
  - Cassandra running in Docker since 2015
  - Multitenant relational DBs in Docker since 2017
    https://www.percona.com/live/18/sessions/containerizing-databases-at-new-relic-what-we-learned
CoreOS NOT CentOS

- Containers containers containers
- Ignition replaces need for Puppet
c.AssertUnit("newrelic-infra.service")

- name: newrelic-infra.service
  enable: true
  contents: |
    [Unit]
    Description=New Relic Infrastructure Agent
    After=docker.service
    Requires=docker.service
    [Service]
    TimeoutSec=0
    Restart=always
    ExecStartPre=-/usr/bin/docker kill %p
    ExecStartPre=-/usr/bin/docker rm %p
    ExecStartPre=/usr/bin/docker pull ...
    ExecStart=/usr/bin/docker run \
    ...

HIGH-LEVERAGE INTERFACES

**TERRAFORM**

- Interface between us and the cloud
- Declarative infrastructure-as-code
- Allows repeatability
- Developed our own Terraform providers as necessary
Second System?
2X

OPERATIONAL LOAD
SAME

TIME TO BUILD A NEW REGION
The Goldilocks balance: Choose the right work
THE PLAN
WE HOPED...

- Discovery
- Fan out
- Test
- Release
Quick ramp-ups
How do you prioritize work?
Autonomous teams
AN EXAMPLE OF PROJECT PRIORITIZATION

1. One team’s must-ship project. Everyone makes sure this team can succeed.

2. High-priority cross-cutting project. All teams do their part as soon as possible.

3. Feature promised to marketing.

4. Future highest-priority project. Don’t block the top, but this needs to ramp up.
QUICK RAMP-UPS

WHAT WASN’T READY?

- Full documentation of what we were asking for
- Service discovery
- Other core tooling
- Easily digestible philosophy to help people make decisions
Prepare for what happens when a project suddenly receives high priority. Produce project philosophy document.

Prioritizing important work across the company
Moving goalposts
MOVING GOALPOSTS

UPFRONT WORK

- Containerization
- Move to Container Fabric (Mesos)
- Service discovery via env vars
MOVING GOALPOSTS

UPFRONT WORK

- Containerization
- Move to Container Fabric (Mesos)
- Service discovery via env vars

LATER WORK

- Replace hardcoded env vars with discovery_path
- base_environment in YAML config
- Regional redirection
Heroes
3

BUILDOUTS
MOVING GOALPOSTS

THE REALITY OF A BUILDOUT

1. A team does the work in the US.

2. Wait a few days or weeks. (In reality, do other things.)

3. Backpack team tries to deploy the team’s services, finds problems.

4. Team is now potentially blocking the buildout.

5. Frustration.

6. GOTO 1.
Steel thread: Validate a design using a subproject that tests it thoroughly.
Avoid waterfall planning. Agile is still good.

Use a small number of teams that form a complete system as a test case.
Be more honest and transparent.

Hidden work, even if only by acknowledging that unknown future work exists.
Communication is hard
COMMUNICATION METHODS

- Blog posts
- Town hall events
- Checklist app
- Linting
- Emails
COMMUNICATION IS HARD

COMMUNICATION METHODS

- Blog posts... don’t get read.
- Town hall events... are optional.
- Checklist app... doesn’t get looked at.
- Linting... doesn’t get used.
- Emails... don’t get read.
Centralized documentation
Put all project requirements and deliverables in one place, with a user-readable changelog (not Git commits).

Have better linting.

Blogging internally.

Communicating using as many channels as possible.
Local maximums
LOCAL MAXIMUMS

YOUR TEAM 3 YEARS AGO

- 20 services

- Want:
  - chained deploys
  - shared configuration
  - service discovery
LOCAL MAXIMUMS

YOUR TEAM NOW

- 20 services
- Have:
  - chained deploys
  - shared configuration
  - service discovery
- Don’t have: the standardized platform that was built in the meantime
Future tooling

Team-specific tooling

Standardized tooling
LOCAL MAXIMUMS

START

Have more empathy for teams stuck in a local maximum. Communicate well in advance, hopefully close gaps early.

STOP

Making assumptions about how teams or individuals will react.

CONTINUE

Making standard tooling better.
Leaning on what you have
LEANING ON WHAT YOU HAVE

IN-FLIGHT PROJECTS

- Container Fabric team building Mesos-based platform
- Build and Deploy Tools building Grand Central build/deploy system
- DB Engineering building Megabase, containerized DB platform
Huge upticks in adoption rate
LEANING ON WHAT YOU HAVE

START
Make clear which priorities are highest for infrastructure teams.

STOP
Look for high-leverage work a small number of teams can do.

CONTINUE

Pilot phase
THE ORIGINAL PLAN

- Discovery
- Fan out
- Test ourselves
- Release
THE REVISED PLAN

- Discovery
- Fan out
- Test ourselves
- Run a pilot
- Release
PILOT PHASE

START

STOP

CONTINUE

Magical thinking.
Where we are now
It worked
EU region
Disaster recovery
Less busywork
95%
SERVICES IN CONTAINER FABRIC
The New Relic product benefits
We’ve learned a lot
A little bit of magical thinking. Trying bold things.
THANK YOU

Twitter: @aughr
micro.blog: @aughr
Mastod on: aughr@pgh.social