



---

# Microservice Trade-Offs

---

Leonid Dinershtein





# Microservices Resource Guide

<https://martinfowler.com/microservices/>

<http://howtcookmicroservices.com/>

# Microservice Trade-Offs



- **Strong Module Boundaries**

Microservices reinforce modular structure, which is particularly important for larger teams.

- **Distribution**

Distributed systems are harder to program, since remote calls are slow and are always at risk of failure.

# Strong Module Boundaries VS Distribution

18 services

17 DB, clusters

13 clients: faraday **http** client

Rabbitmq queues














Cache !

**God gem**

common\_gem v. \*5.21.0\*

18 pull requests [OPEN](#)

## Repository

	rosi_promotion_client
	rosi_catalog_client
	rosi_oms_client
	rosi_faraday_client
	rosi_delia_client
	rosi_subscription_client
	rosi_my_business_client
	rosi_payment_management_client
	rosi_credits_client
	rosi_rewards_client
	rosi_jive_client
	rosi_pulse_client
	rosi_configuration_client

# Microservice Trade-Offs



- **Independent Deployment**


Simple services are easier to deploy, and since they are autonomous, are less likely to cause system failures when they go wrong.

- **Eventual Consistency**

Maintaining strong consistency is extremely difficult for a distributed system, which means everyone has to manage eventual consistency.

# Independent Deployment VS Eventual Consistency

 rosi\_rewards\_service [feature/RONX-212](#)

 rosi\_subscription\_service [feature/RONX-212](#)

rosi\_tax\_service [master](#)

common\_gem v. \*5.21.0\*



**deploy train** APP 7:57 PM ☆

rosi\_credits\_service merged to release. [View Pull Request](#)

rosi\_cart\_service merged to release. [View Pull Request](#)

rosi\_oms\_service merged to release. [View Pull Request](#)

rosi\_email\_service merged to release. [View Pull Request](#)

rosi\_catalog\_service merged to release. [View Pull Request](#)

rosi\_delia\_service merged to release. [View Pull Request](#)

rosi\_inventory\_service merged to release. [View Pull Request](#)

rosi\_mockup\_service merged to release. [View Pull Request](#)

rosi\_my\_business\_service merged to release. [View Pull Request](#)

rosi\_promotion\_service merged to release. [View Pull Request](#)

# Microservice Trade-Offs



- **Technology Diversity**

With microservices you can mix multiple languages, development frameworks and data-storage technologies.

- **Operational Complexity**

You need a mature operations team to manage lots of services, which are being redeployed regularly.

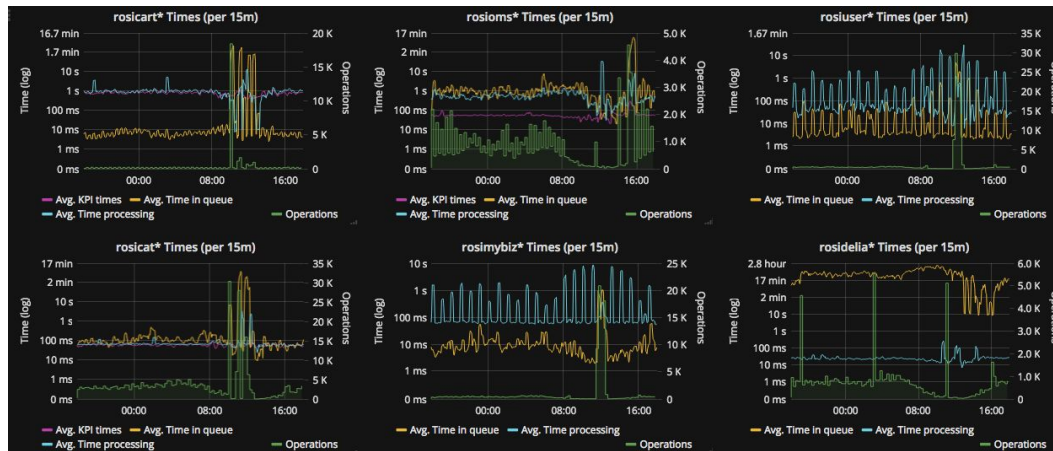
# Technology Diversity VS Operational Complexity

Rails 4 & Rails 5

Ruby & Java

Elastic Search 2 & Elastic Search 5

Redis & Memcached



Name
rosi_user_service
rosi_ugc_service
rosi_catalog_service
rosi_cart_service
rosi_my_business_service
rosi_inventory_service
rosi_oms_service
rosi_subscription_service
rosi_mockup_service
rosi_payment_management_service
rosi_promotion_service



# Questions?

Leonid Dinershtein

[leonid@whitespectre.com](mailto:leonid@whitespectre.com)

