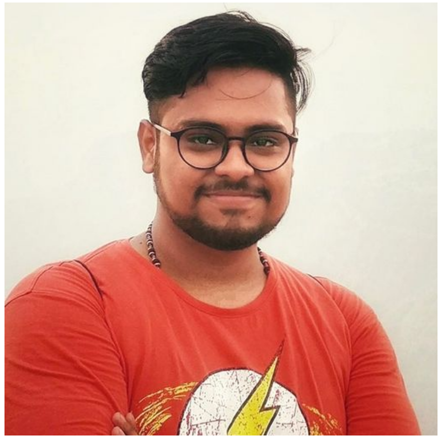
Start Small, Scale Big

Building and Scaling Platforms and SRE Culture at Startups

Yash Shanker Srivastava

- 7 years in the industry
- based in Bangalore, India
- Engineering Manager at Toplyne
- started with developing payment products at PayU Payments, India
- DevOoopsed for companies like Setu, Validus Capital, 2c2p and Toplyne
- love to contribute to design and development of Distributed Systems



devooops.dev
srecon@devooops.dev
devooops.dev/linkedin
devooops.dev/twitter
devooops.dev/github

Introduction

Site Reliability Engineering

Site reliability engineering (SRE) is **the practice** of using software tools to **automate IT infrastructure tasks** such as system management and application monitoring.

Organizations use SRE to ensure their **software applications remain reliable** amidst frequent updates from development teams.

SRE especially **improves the reliability of scalable software systems** because managing a large system using software is more sustainable than manually managing hundreds of machines.

Platforms Engineering

Platform engineering is the discipline of **designing and building Internal Developer Platforms**, toolchains and workflows that **enable self-service capabilities for software engineering organizations**.

Platform engineering teams **apply software engineering principles** to **accelerate software delivery**.

Platform engineers **ensure application development teams are productive** in all aspects of the software delivery lifecycle.

Startup vs ScaleUp vs Large Org

Startup	**Scaleup**	**Large Org**
5-20 Engineers	50-200 Engineers	above 500 Engineers
Developing its first product | Achieving PMF	Rapid growth for the product/s | Achieved PMF	Multiple proven products
Trying to establish business model	Proven business model, now scaling it	Matured and scaled business model
No revenue | Pre-seed to Series-A	Some revenue | Series B to Series D	pre-IPO or Public
2-15 mil USD in bank	Above 50 mil USD in bank	lol amount in bank

Goals : Startup vs Scaleups

- **Startup**
    - Develop the product and find PMF
    - Go to market and onboard first few customers
    - Absorb feedback from the customers and rapidly build for them
    - Be frugal, but okay to spend
- **Scaleup**
    - Stabilise the product
    - Scale efficiently without breaking the product
    - Build stable features
    - Be frugal, but spend efficiently

Reliability & Platform Engineering

- automation
- shared philosophy of DevOps
- stability and scalability
- collaboration
- continuous feedback and improvements

TLDR ; Make others’ life easier.

Imbibing SRE Philosophies in an Engineering Org

Alignment of platforms with the needs of the business

- serve tech to solve business
- save money
- not get influenced by industry trends
- avoid fancy technology: you don't need it, and it's slow to use
- work through the constraints
- make Engineering sustainable

Staying adaptable while scaling

- make an Engineering Charter, that defines your culture
  - Separate SRE/Platforms charter, while scaling out
- take short-term decisions, keeping long term vision in mind
- tech-stacks, principles and philosophies of the SRE culture need to be adaptable
- achieve business excellence immediately, but tech excellence in iterations

Example :

- use saas and managed services
- start with a monolith
- dockerize your services, but don't use k8s
- use SaaS o11y provider, but with otel-exporters
- use infrastructure-as-code
- implement continuous delivery

Setting up SRE culture from the first Engineer

- gatekeeping bullshit out of Engineering culture is important
- ensure that reliability remains at the core of the organization's culture as it scales
- it is difficult in the growth phase to start with standardization platforms
- retrofitting of SRE principles adds toil
- no-SRE model (thanks PSN team 😉) works wonders
  - implement the philosophy, not the role/designation/team
  - ones who write the code, [must/should] know best how/where to run it

Strategies for Developing SRE Platforms in Constraints

Adapting SRE philosophy based on constraints

- time
  - GTM being the goal, there isn’t much time to get to tech eutopia
  - ship or die
- money
  - frugality (current market conditions, well. 😢)
  - no money to waste
- people
  - people come with biases and practices
- knowledge
  - team members have different knowledge levels and learnings

Decisions regarding suitable technologies and services

Decisions regarding suitable technologies and services

cool and best is not the always the most suitable.

Decisions regarding suitable technologies and services

- monolith vs microservices ⏱️
- self-hosting vs managed services 💲
- binaries/executables vs containers vs functions 🛠️
- docker-compose vs Beanstalk vs ECS vs EKS
- datadog vs ELK vs Signoz
- real-time/sync vs event-driven/batch
- database
- SaaS vs in-house self hosted

Tldr: building business logic vs building infra

Synergy between Platform Engineering, DevOps, and SRE Principles in Practice

- shared goals
- collaboration and cross-functional work
- DevOps in their foundation
- automation and standardization
- continuous feedback and improvements

TLDR ; Make others’ life easier.

Open Source your SRE and Platforms

(within your Engineering team)

It’s DAY 2

It’s DAY 2

Where the fudge is

SECURITY ?

It´s DAY 2

Where the fudge are

SECURITY ?

COMPLIANCE ?

It’s DAY 2

Where the **fudge** are

SECURITY ?

COMPLIANCE ?

SECOPS ?

Case Study 1 : Setu

SETU

API banking infrastructure for India's financially-excluded millions

Our mission

We want to accelerate India's GDP growth by making digital financial services affordable and accessible to all.

Our vision

APIs for anything fintech—and that's the only way to scale.

March 2020 vs May 2022

- 1st product (collect-payments) went live
- 0 revenue. USD 15mil in bank.
- 3 micro-services
- a few thousand API requests per month
- a few GBs of data processing per month
- 1 team of total 5 engineers
- 1 Platform Engineer

- 8 live products across, payments, investments, lending and KYC
- over 50 microservices in production
- ~25 million API requests per month
- ~50 TBs of data processing per month
- 8 teams of over 30 engineers
- 2 Platform Engineers

How we started at Setu ?

- March 2020
  - 3 microservices deployed across 1 live product.
  - deployed on AWS
  - containerized services and deployed on Elastic Beanstalk
  - infrastructure managed in Terraform
  - DevOps Team responsible for Infra changes, security, compliance and peripherals
  - manual deployments
- May 2020
  - Launching the second product
- June 2020
  - Plans to launch 5 more products rapidly within 6 months

How we started at Setu ?

- deployments involved downtimes
- change management and patching was challenging
- lack of standardization
- writing new infrastructure was boring and time taking
- multiple regulatory audits
  - api security testing
  - recurring full infra vapt audits
- decentralised flow of architectural decisions were difficult to standardise

Birth of Platform Engineering at Setu

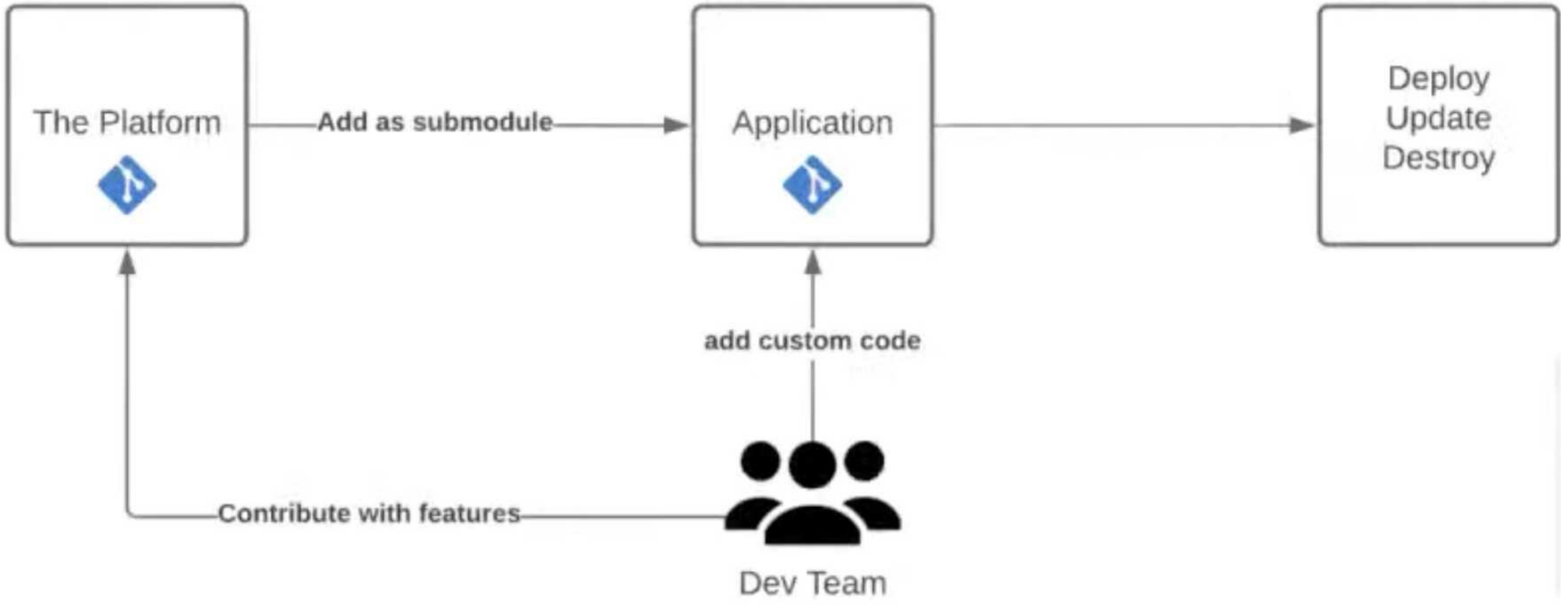
Standardised DevOps Platform to be used across Setu

- modularised self-serve infrastructure as code in Terraform
  - ECS based containerized deployments
  - RDS and MongoDB Atlas
  - Site-site VPN with multiple bank infrastructures
- feature packed hardened Packer based AMIs
  - HIDS, NIDS, FIM, SSH Hardening, antivirus
- ELK based o11y for logs and metrics (no traces)
  - alerting, incident management on PagerDuty and Slack
- built in Security and Compliance
  - ISO 27001:2013 , RBI SAR, Banks and NPCI audits audit checks
  - Cloud Security Alliance - Star Level One certified

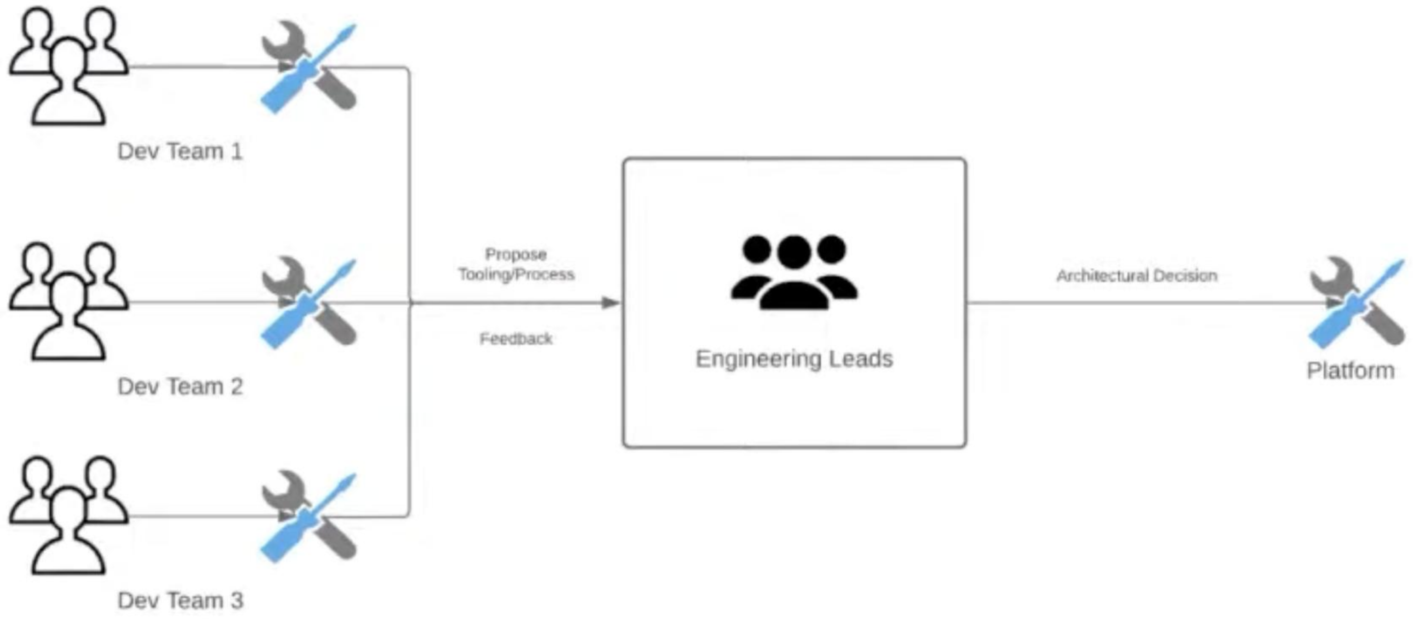
May 2022

- 8 live products across, payments, investments, lending and KYC
- over 50 microservices in production
- ~25 million API requests per month
- ~50 TBs of data processing per month
- 8 teams of over 30 engineers
- streamlined incident management and 3 level on-call rotation policies
- 2 Platform Engineers

How Dev teams used the Platforms



How architectural decisions flow into the Platform ?



Impact

- Go to Production depended only on writing business logic for the product.
- All SRE components built into the Platform from Day 0 of dev.
- Dev Teams owned end to end infrastructure and operations of their product.
- Setu had to go undergo 20 InfoSec and Compliance Audits in a single year
  - All done seamlessly with minimal dev effort
- PR to release time gone down to 10 minutes
- Standardised infra and practices across the business.

Case Study 2 : Toplyne

Toplyne

The platform that Cloudflare, Vercel, and Canva use to find sales pipeline in their self-serve funnel

Toplyne helps you **build sales pipeline** from your self-serve account base.

Our AI combines your **product usage & CRM** information with third-party enrichment, to push the right accounts into your **existing CRM**...

..with no data science & engineering favors.

Canva

Millions of people around the world use Canva to achieve their goals. It's important that we're able to identify users where our premium subscriptions are the right fit. Toplyne enables us to effectively target and scale these efforts at our speed of growth.

David Burson,
Head of Revenue, Canva

June 2022 vs June 2023

- product is live
- monolith, with few data services spread across lambda functions and in Airflow
- a few thousand API requests per month
- 10s of TBs of data processing per month
- 5 Data Scientists, 3 Data Engineers, 5 FullStack Devs
- no “standardised” infra-as-code
- SRE practices instilled from the 1st Engineer
- automated deployments
- 1 Platform SRE

- product is live
- still a monolith, tens of services across Lambda, Batch, Airflow
- 100s of TBs of data processing per month
- 5 Data Scientists, 4 Data Engineers, 10 FullStack Devs
- **standardised infra-as-code platform**
- continuous delivery
- devs own their ops
- 2 Platform SREs

State of Platforms in June 2023

- no Kubernetes [🤷🏻‍♂️]
- prefer native AWS services [save time]
- migrated Airflow from MWAA (AWS managed Airflow) to self-hosted [more control]
- migrated o11y from Datadog to self-hosted Grafana LGTM stack [save costs]
- platform team produces self-serve infra and services for DS, DE and Product teams
- regular knowledge sharing in a fast moving org [sustain the culture]
- push to release takes 8-10 minutes (ci, secops, infra-updates, cd)
- platform is the product of the SRE team
    - support in the form of new features, not “support”

Why “How Google or Netflix Do SRE” is Not a Good Place for Startups to Start With

Why How Google or Netflix Do SRE is Not a Good Place for Startups to Start With

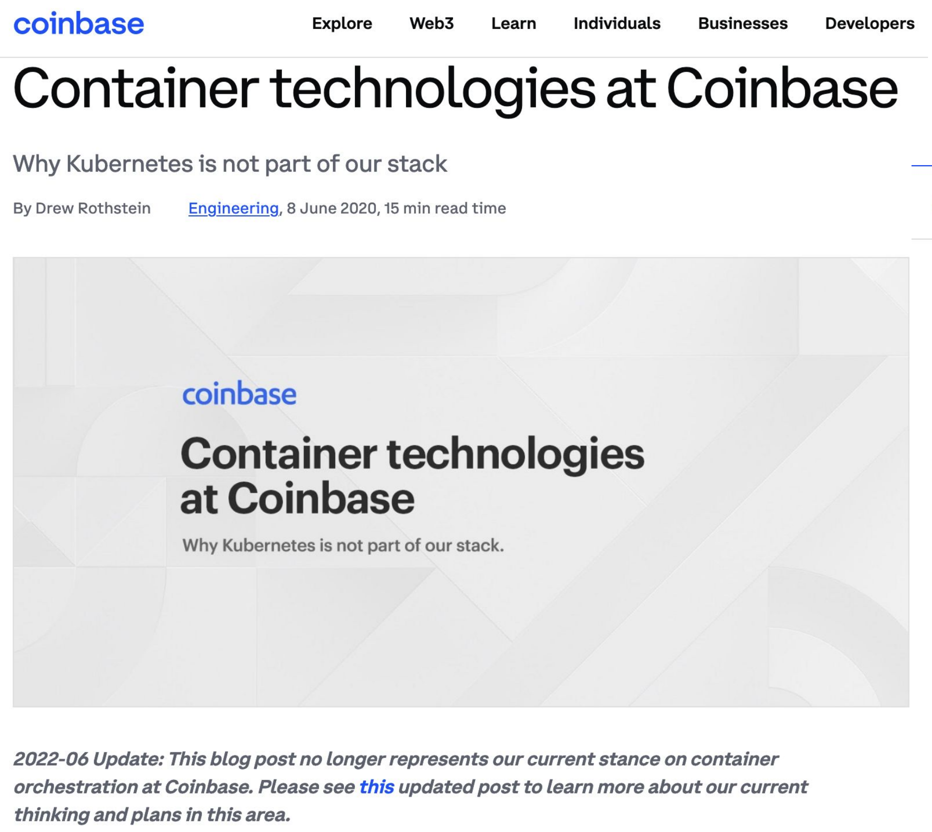
- requirements
  - business
  - tech
- scale
- maturity
  - org
  - engineering team
- budgets

Conclusions

Framework for setting up a Culture of Site Reliability Engineering

- implement principles, not the role/team
- standardise, yet allow extensibility
- stay adaptable
- prioritise business needs over exciting tech
- iterate towards perfection, but don't add debt
    - the perfect tool/practice/culture doesn't exist
- use plug/play managed services to start with
- prefer productization, over support based service
- just get shit done

DON'T GET EXCITING IDEAS FROM CONFERENCES/PODCASTS/YT VIDEOS, AND RUSH TO IMPLEMENT IN YOUR ORG. 🤪



coinbase Explore Web3 Learn Individuals Businesses Developers

Container technologies at Coinbase

Why Kubernetes is not part of our stack

By Drew Rothstein Engineering, 8 June 2020, 15 min read time

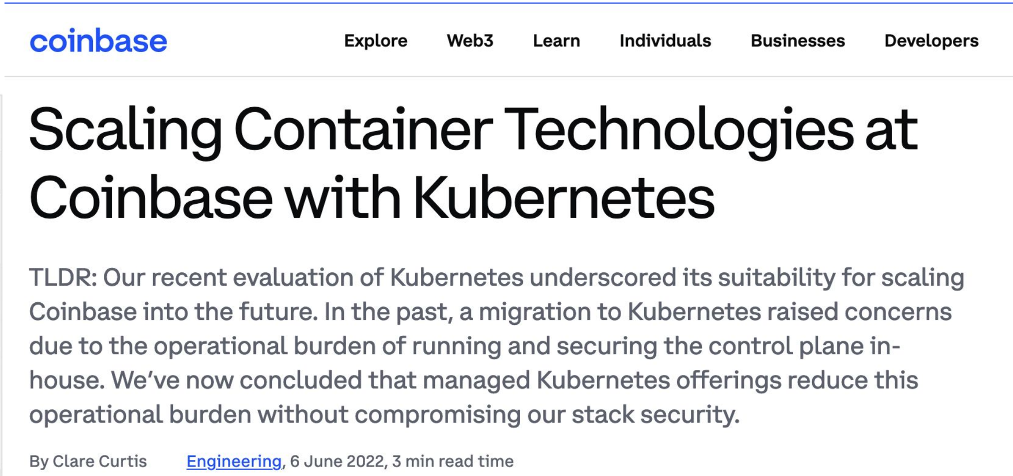
coinbase

Container technologies at Coinbase

Why Kubernetes is not part of our stack.

2022-06 Update: This blog post no longer represents our current stance on container orchestration at Coinbase. Please see this updated post to learn more about our current thinking and plans in this area.

June 2020



coinbase Explore Web3 Learn Individuals Businesses Developers

Scaling Container Technologies at Coinbase with Kubernetes

TLDR: Our recent evaluation of Kubernetes underscored its suitability for scaling Coinbase into the future. In the past, a migration to Kubernetes raised concerns due to the operational burden of running and securing the control plane in-house. We've now concluded that managed Kubernetes offerings reduce this operational burden without compromising our stack security.

By Clare Curtis Engineering, 6 June 2022, 3 min read time

June 2022

Questions ?