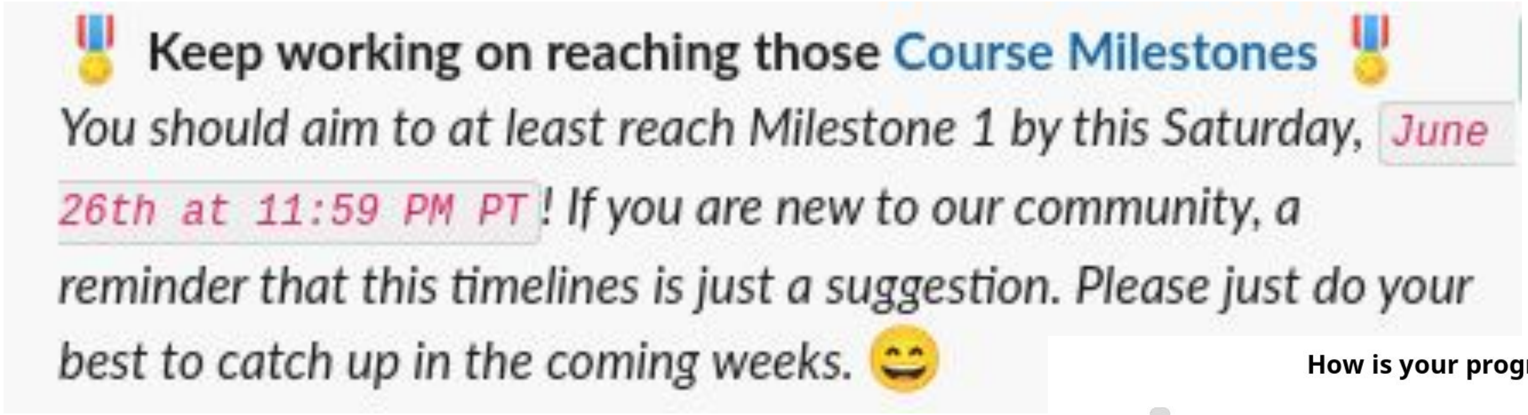
# Memers 2nd Meeting

SUSE Cloud Native Scholarship - Udacity

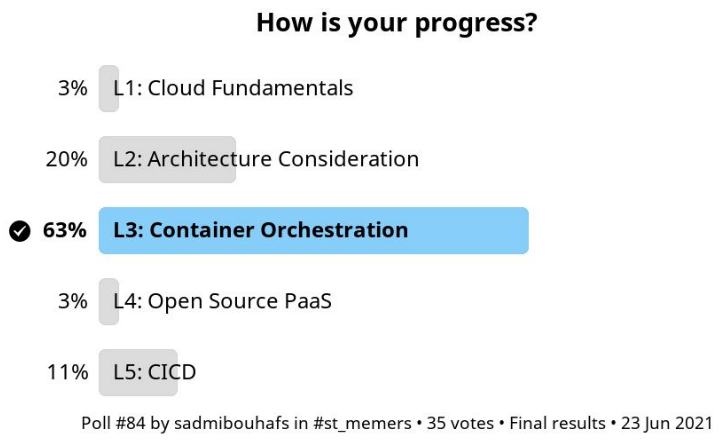
## Agenda

- Announcement
- Memers activities
- Monolith vs Microservices
- Q&A session
- Fun Time

## Announcement



🎖️ Keep working on reaching those Course Milestones 🎖️
*You should aim to at least reach Milestone 1 by this Saturday,* `June 26th at 11:59 PM PT`*! If you are new to our community, a reminder that this timelines is just a suggestion. Please just do your best to catch up in the coming weeks.* 😄



**How is your progress?**

| | |
|---|---|
| 3% | L1: Cloud Fundamentals |
| 20% | L2: Architecture Consideration |
| ✓ **63%** | **L3: Container Orchestration** |
| 3% | L4: Open Source PaaS |
| 11% | L5: CICD |

Poll #84 by sadmibouhafs in #st_memers • 35 votes • Final results • 23 Jun 2021

## Weekly quizzes

Organized by:

Yojana, Memphis & Vedita.Kamat

Link: https://quizizz.com/join?gc=35777030

quizizz.com

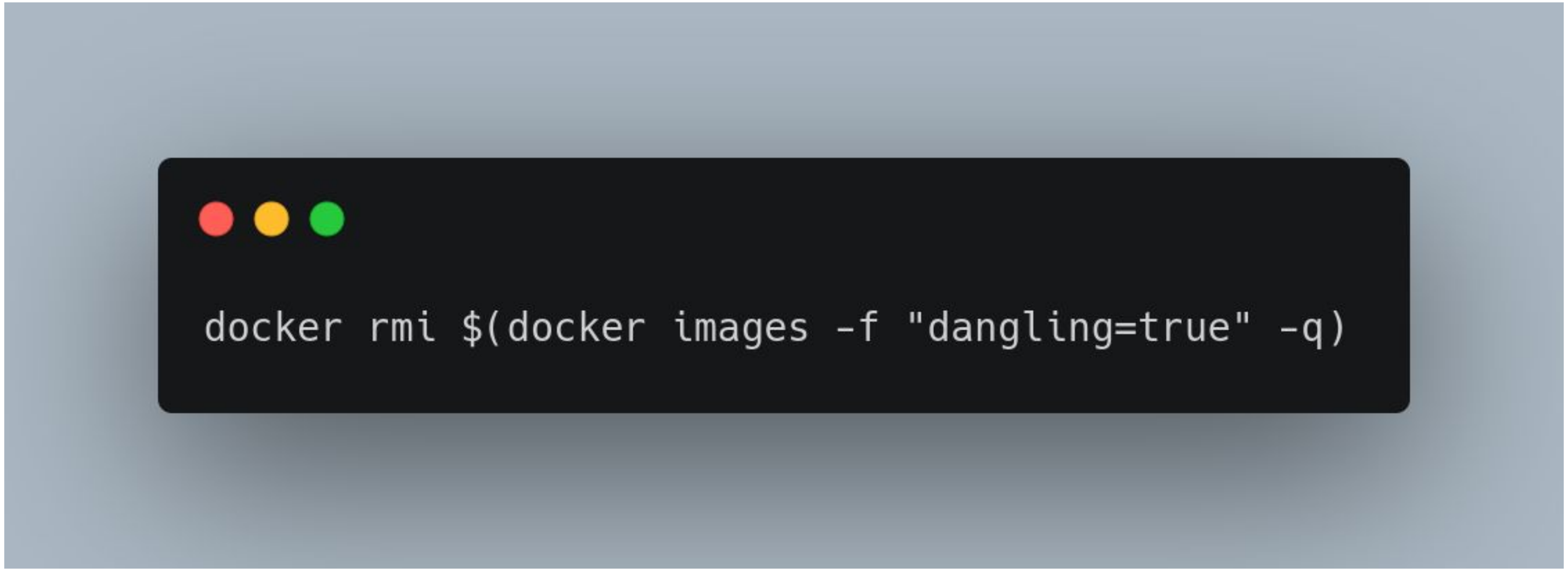
Quizizz: You've been invited to a Quizizz game

Click the link to join now. (23 kB)

## Meme War

Stay Tuned tomorrow Sunday 27th, organized by:
Biswajit & Verrah

## DevOps Tips



```
docker rmi $(docker images -f "dangling=true" -q)
```

## Tell it with a meme

Organized by :
Dimitra.Karamperi,
Verrah, Ghano,
Vijay.Bhargav.Reddy

why you are explaining serious concepts with memes
Memes are better than text

## Memers activities

- **Projects:**

  Charlie.Tseng, Mayur.Kanojiya, Samah.Anemiche,  & Ebinbin.Ajagun

- **Blogathon:**

  Vedita.Kamat, kumsomi & Manasvi.Trivedi

- **Tech sessions:**

  Nitin Bodke & Sadmi

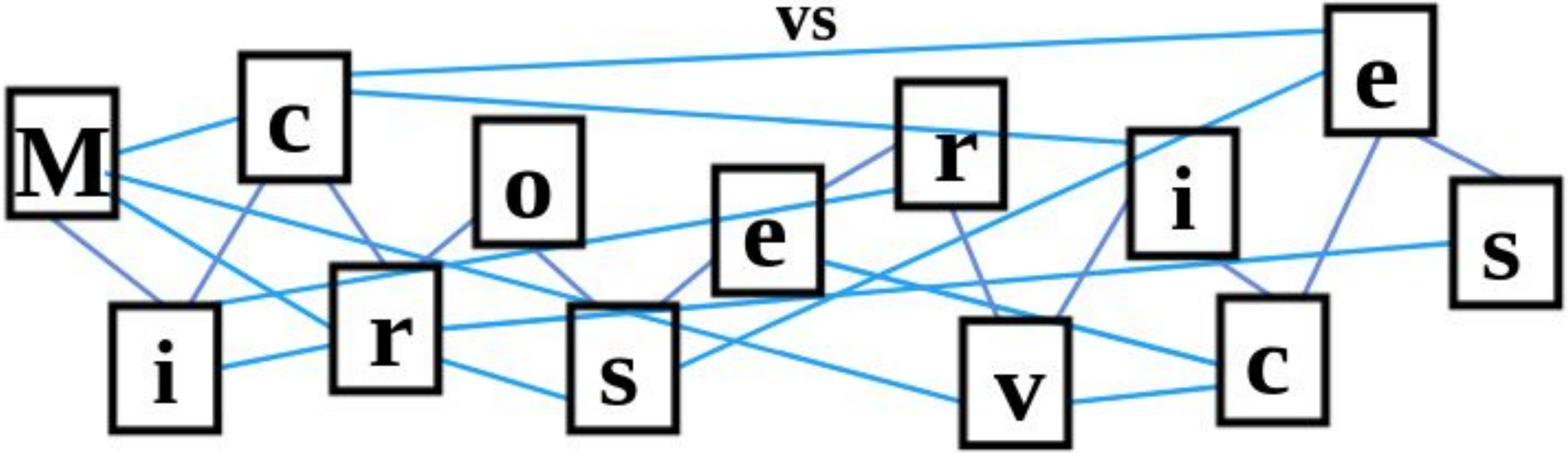
- **Cloud Native Memers Ebook:**

  Vijay.Bhargav.Reddy & Sadmi

## Monolith vs Microservices

## Monolith

vs



## Suppose you start your own Project, your own Startup, your goals are:

- Express your idea to the public (the part of your service is almost always related to web-services)
- Launch it as quick as possible

So, for rapid deployment you need:

- your coding experience
- your favorite coding stack (PHP/Ruby/Python/JavaScript/etc.+HTML/CSS)
- your favorite Database software (MySQL/PostgreSQL/etc.)
- the place where it all would reside (shared hosting/VPS, AWS/GCP/Azure/etc.)

>>> It’s a typical **MONOLITH** architecture, so you don’t need more at the moment

## Hoorah!!! You launched it, you did it, it becomes more popular!!! How lucky you are!!!

What you have for today:

- Everything resides on a single server (application+database = typical LAMP/LEMP stack)
- You decide to hire additional developers to support and extend your codebase with your planned features
- You noticed the significant load to your platform, performance issues faced, you should somehow solve it
- Contacted your hosting provider’s Support Team, opened a ticket
- Support Team would rather advice you upgrade your plan

## Illusion of a solved problem

- You changed your plan to one with much bigger quantity of CPU cores, bigger RAM size, chose SSD as a storage backend
- It works smoothly for now

But in the meantime

- You start thinking of how to gauge and monitor your app's performance
- You can observe only the main app's process with its forked children consuming resources – a bit hard to say which part of subsystem is eating the most
- You're a not yet efficient in profiling and debugging

## Subsequent growth

- Your app is being enriched with additional functionality
- Newly released feature updates are much slower to release for production -> inconveniently maintainable pull request queue in a one repo, or even worse – each developer has a committing privilege to the master branch
- Codebase becomes more error prone

You think of 4 aims now:

- Implement Agile practices in your Dev Team
- Plan to refactor your codebase
- Hire/promote the current staff member to Software Architect
- Hire/promote the current staff member to SRE/DevOps Engineer

## Here it comes…

## Decision of breaking the Monolith to Microservices

## Software Architect

- Performs current architecture analysis
- Builds a new software design

  >>> decision of breaking the Monolith to Microservices is made here

## SRE/DevOps Engineer

- Configures his/her environment, orchestrating tools
- Prepares new Git repos, branches
- Deploys and configures container registry
- Prepares sandbox and infrastructure platform for hosting an app pieces
- Creates and configures CI/CD pipelines
- Deploys monitoring services and logs collector (the visibility of what is working where and how is enabled)

## Dev Team

- Divided into groups of functionalities
- Each group selected the person to be the group Lead, who is also is in charge of repo commitments
- Codebase refactoring succeeds
- Testing and staging phase completed
- Logging, metrics and tracing options were also implemented
- Did a significant amount of work, weeks of time spent ;-)

## Transition to the new micro-serviced architecture performed

## to recap…

## Pros of Microservice software architecture

**With that implemented:**

- metrics, logging, tracing options in your code
- well-designed API interface architecture between services
- testing/staging environment
- inter-teams collaboration
- CI/CD pipelines
- k8s cluster

**You would definitely benefit from:**

- deep oversight of what is going on at every layer
- subsequent upgrades and maintenance, which are being performed seamlessly now
- each separate service is developed, tested, staged, pulled to production separately
- smart update/rollback technics (rolling, canary etc.) of your Kubernetes compatible platform
- performance advantage of automatic scaling and recovery

## Cons

- alongside coding your main logic you should embed your code with additional functionality
- design of optimal API interface between services
- not simple to implement at your first

# Thanks for attention!!!

SUSE Cloud Native Foundations Scholarship Program

Prepared by: @Muhammet.Berdimyradov

Inspired by: @Sadmi

#st_memers

## Fun Time

I used a recursive function
You didn't forget the exit condition, right?
I used a recursive function
You didn't forget the exit condition, right?

Wheres my fucken money?

## Fun Time

made with
REFACE APP
I'm going to change the world with deep learning
For the better, right?
For the better, right?