Site Reliability Engineering for Cloud - CMPE 189

Fall 2026 | In Person | 3 Unit(s) | Mon/Wed 1:30-2:45

Charles W Davidson College of Engineering - Computer Engineering

Contact Information

Instructor: Ben Reed (designed by Salim Virji at Google)

Office hours for Spring 2026:

TBD

Course Description and Requisites

Hands-on introduction to site reliability engineering (SRE). Topics include datacenter infrastructure, virtualization, storage, and scalability. Application lifecycle management from build to deployment to observability. Capacity management. Cross-application security. Hands-on projects using industry best practices, RPC frameworks, and observability tools.

Prerequisite: Instructor consent

Letter Graded

Course Learning Outcomes (CLOs)

Upon successful completion of this course, students will be able to:

1. Deploy and manage applications in datacenters.
2. Identify essential metrics to *measure* service performance
3. Create and deploy automation to maintain system performance.
4. *Develop* Service Level Objectives based on program-level metrics.
5. Use post-mortems to improve systems after failures.

Course Materials

Textbook

[1] B. Beyer, N. R. Murphy, D. K. Rensin, K. Kawahara, and S. Thorne, *The Site Reliability Workbook: Practical Ways to Implement SRE*. Sebastopol, CA: O'Reilly Media, 2018. [Online]. Available: https://sre.google/workbook/table-of-contents/

Supplemental Texts

[2] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, *Site Reliability Engineering: How Google Runs Production Systems*. Sebastopol, CA: O'Reilly Media, 2016. [Online]. Available: https://sre.google/sre-book/table-of-contents/

[3] H. Adkins, B. Beyer, P. Blankinship, P. Lewandowski, A. Oprea, and A. Stubblefield, *Building Secure and Reliable Systems*. Sebastopol, CA: O'Reilly Media, 2020. [Online]. Available: https://google.github.io/building-secure-and-reliable-systems/raw/toc.html

[4] C. Majors, L. Fong-Jones, and G. Miranda, *Observability Engineering: Achieving Production Excellence*. Sebastopol, CA: O'Reilly Media, 2022. [Online]. Available: https://learning.oreilly.com/library/view/observability-engineering/9781492076438/

[5] M. Kleppmann and C. Riccomini, *Designing Data-Intensive Applications*, 2nd ed. Sebastopol, CA: O'Reilly Media, 2026. [Online]. Available: https://learning.oreilly.com/library/view/designing-data-intensive-applications/9781098119058/

[6] L. A. Barroso, U. Hölzle, and P. Ranganathan, *The Datacenter as a Computer: Designing Warehouse-Scale Machines*, 3rd ed. San Rafael, CA: Morgan & Claypool, 2018.

Articles

[7] A. Verma, L. Pedrosa, M. R. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at Google with Borg," in *Proc. European Conf. Computer Systems (EuroSys)*, 2015. [Online]. Available: https://research.google/pubs/pub43438/

[8] D. K. Rensin, "Kubernetes - Scheduling the Future at Cloud Scale," O'Reilly Media, 2015. [Online]. Available: https://research.google/pubs/pub43826/

[9] R. Ananthanarayanan *et al.*, "Photon: Fault-tolerant and Scalable Joining of Continuous Data Streams," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 2013. [Online]. Available: https://research.google.com/pubs/pub41318.html

[10] M. Pandey, "Building Netflix's Distributed Tracing Infrastructure," *Netflix Technology Blog*, Oct. 2020. [Online]. Available: https://netflixtechblog.com/building-netflixs-distributed-tracing-infrastructure-bb856c319304

[11] Twitter Engineering, "Deterministic Aperture: A distributed, load balancing algorithm," *Twitter Engineering Blog*, 2019. [Online]. Available: https://blog.twitter.com/engineering/en_us/topics/infrastructure/2019/daperture-load-balancer.html

[12] D. McNutt, "Distributing Software in a Massively Parallel Environment," presented at *USENIX LISA '14*, Nov. 2014. [Online]. Available: https://www.usenix.org/sites/default/files/conference/protected-files/lisa_2014_talk.pdf

Other technology requirements / equipment / material

Programming assignments will be a significant part of this course, so access to a computer is required. Students will become familiar with the Go programming language, which we will use for the code and development in this course.

Course Requirements and Assignments

Unauthorized help and AI Assistants

SJSU does not have a department- or school-wide policy on the use of AI in classwork. This class encourages students to write and debug on their own, without the use of AI tools.

Programming Assignments

Students will begin with the source code to microblogging software, and will expand and extend this over the course of the semester.

- Course meets twice each week for 75 minutes, of which 45-50 minutes direct instruction plus engagement through iClicker.
- Exercises represent hands-on, take-home / lab assignments.
- There will be a quiz related to each assignment. Some quizzes will be for more than one assignment. The quiz will be simple coding questions related to the assignment. If your assignment score is more than 20% of the related quiz score, the assignment score will be adjusted so that it is not.

Class Preparation

To encourage preparation for class, an online chapter quiz will review the concepts for the upcoming class covered in the reading material. Students can use the reading material as

references and can take the quiz multiple times, but the quiz must be done individually and using only the reading material.

iClicker is also used to stimulate discussions in class. It cannot be done remotely. Since students may occasionally need to miss class due to unforeseen circumstances or other commitments, 70% participation score will be logged as 100% in the gradebook. In other words, you can miss 30% of iClicker and still get 100%. Scores below 70% are prorated and you cannot get more than 100% for the iClicker component of your grade.

Grading Information

I do not grade on a curve. The exams and assignments measure what you are expected to have learned. We design exams so that everyone who masters the material in the study guides can get 100%.

There aren't many opportunities for extra credit. We have three exams (including the final); missing an exam will result in an assigned grade of 50% for the exam.

The grade for the class is based

component	**weight**
midterm 1	20%
midterm 2	20%
final	20%
assignments	15%
quiz	15%
chapter quizzes	5%
in-class iClicker	5%

This class uses minimum grading: you cannot get below a 50% on any submission or exam. For example, if you do not submit a solution or your submission falls far short and only scores 35%, you will be assigned a 50% in the grade book. **The minimum grading does not apply to cases of academic integrity.**

Percentage	Grade
96 and above	A+
93-95	A
90-92	A-
86-89	B+
83-85	B
80-82	B-
76-79	C+
73-75	C
70-72	C-
66-69	D+
63-65	D
60-62	D-
59 and below	F

University Policies

Per University Policy S16-9 (PDF), relevant university policy concerning all courses, such as student responsibilities, academic integrity, accommodations, dropping and adding, consent for recording of class, etc. and available student services (e.g. learning assistance, counseling, and other resources) are listed on the Syllabus Information web page. Make sure to visit this page to review and be aware of these university policies and resources.

Course Schedule

Week	Topic	Exercise
1	Introduction: Course Purpose, Objectives, Structure. Define distributed systems. What you will learn in this course.	—

Week	Topic	Exercise
2	Datacenters and Physical Infrastructure. Physical environment (power, cooling, security). Hardware architecture (racks, blades, storage, interconnects). Failure domains and fault tolerance. Resource abstraction: from bare metal to virtualization.	Infrastructure modeling: define a model datacenter, including racks, PDUs, and servers.
3	Compute: from single-processor to multi-processor to Cloud. Threading, multi-threading. Inter-process communication (IPC). From monolith to microservices. Containers and container orchestration.	Run in a container; run in multiple instances.
4	Communication: networking topology. LANs vs WANs. Datacenter topologies (fat tree, spine-leaf, et al.). Network metrics.	Client-server program with configurable delay.
5	Storage and Storage Patterns. Storage types, benefits, comparison. Encryption at rest. Networked storage (NAS, SAN). Distributed storage (features, challenges, replication). Data caching.	Implement a key-value store with an API.
6	Data, consistency, and consensus. ACID. NoSQL. Trade-offs and CAP theorem. Consistency types. Consensus (Paxos, Raft, Zab).	Data modeling; extend key-value store to replicate data across multiple nodes.

Week	Topic	Exercise
7	Network Protocols and RPCs. TCP, UDP. HTTP/2. Encryption in transit. Network serialization formats.	Implement a data-transfer function using raw TCP; compare with HTTP.
8	Message-Passing and RPCs. Stubs, best practices (timeouts, retries), frameworks. Pub/Sub.	Use a standard RPC library to create a service interface definition; implement a remote function call.
9	Failure Modes. Fallacies of distributed systems. Failure classification (crash, omission, performance, Byzantine). Resilience patterns: Timeouts, Retries, Jitter/Exponential Backoff, Bulkheads, Circuit Breakers. Cascading failures, thundering herds.	Add retry mechanism.
10	Scaling. Horizontal and vertical scaling. Stateless services. Sharding, partitioning. Caching revisited. Auto-scaling.	Implement a software load balancer.
11	Deployments: Builds, Canary, Rollouts, Rollbacks. Build systems (Bazel). CI/CD. Deployment and canary strategies. Feature flags.	Canary configuration.
12	Identifying and Setting Service Level Objectives (SLOs).	—
13	Measuring Reliability: Instrumentation, SLOs, Observability. Metrics, logs, traces.	Instrument the service to provide structured logs and query-able metrics.

Week	Topic	Exercise
14	Serving Live Traffic, Load Balancing. Load balancer types and algorithms. Health checks. Rate limiting and throttling.	Implement a rate-limiting sidecar.
15	Troubleshooting. Debugging distributed systems (logs, traces, spans). Agentic AI. Incident management. Post-incident reports / post-mortems. Identity: authentication vs. authorization, identity providers, token-based authentication, service authorizations.	Create an endpoint that validates a signed token and grants access.