Zulip Project Structure

The threaded team chat app. A massive Python/Django codebase known for its high code quality, custom frontend, and unique threading model.

Updated 2025-12-30

#zulip #python #django #tornado #typescript #real-time #chat

PNG PDF Copy Prompt

Project Directory

```
zulip/
  zerver/  Main Django App
    models.py  Data models (St…
    views/  API Endpoints
      messages.py
      users.py
    actions/  Business Logic …
    tornado/  Real-time push …
  web/  Frontend Web App
    src/  TypeScript sour…
    templates/  Handlebars temp…
    styles/  CSS
  templates/  Server-side tem…
  puppet/  Infrastructure …
  manage.py  Django CLI
  requirements.txt  Python Deps
```

Repository Info

Repository - zulip/zulip

Stars - 20k+

License - Apache-2.0

Last Analyzed - December 2025

Tech Stack

Backend - Python (Django)

Real-time - Python (Tornado)

Frontend - TypeScript (Custom)

Database - PostgreSQL

Cache - Redis + Memcached

Queue - RabbitMQ

Architecture Notes

Zulip uses a hybrid architecture. **Django** (`zerver`) handles the REST API and database interactions. **Tornado** runs purely as an event queue bridge, holding open long-polling/WebSocket connections to push events from RabbitMQ to clients. This allows Django to remain synchronous and simple.

Key Directories

zerver/ - The core Django application. It contains an `actions/` directory which acts as a 'Service Layer', encapsulating business logic separately from Views.

web/src/ - The frontend is a single-page app but *not* React/Vue. It uses a custom architecture built on jQuery and Handlebars, now migrated to TypeScript. It's optimized for rendering massive message lists.

puppet/ - Zulip includes its own configuration management. The `zulip` Puppet module defines exactly how to deploy the server, enforcing a strict production environment.

Why This Structure?

Zulip is a masterpiece of optimization and code organization. It handles thousands of concurrent users and millions of messages efficiently. Its 'Service Layer' pattern in Django (`zerver/actions`) is a model for how to structure large Python web apps.