eng.minimal_erlang_application_cluster

Alexandr Kirilov (https://github.com/alexandrkirilov)

Minimal Erlang Application cluster configuration.

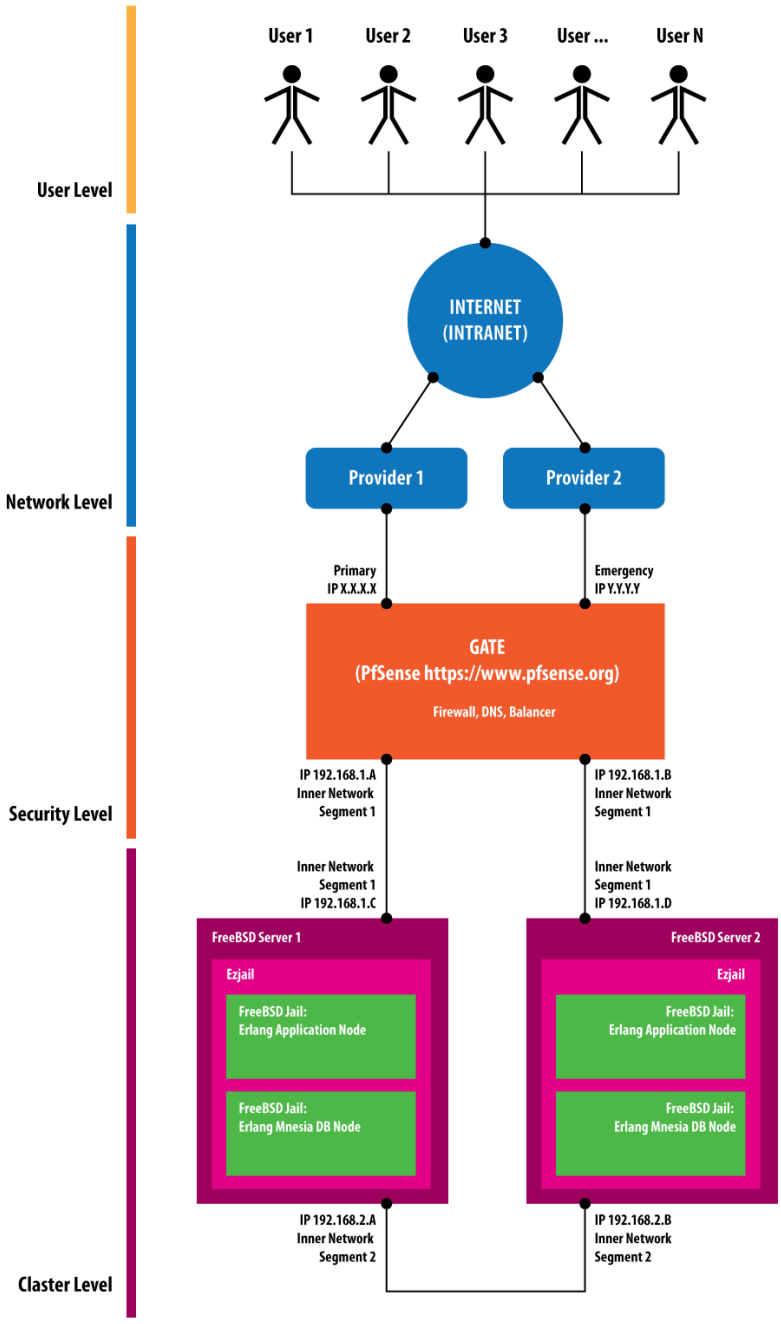
This example of Erlang Distributed Application based on realised different projects from from various fields (security, education, healthcare, games and monitoring) when the budget restricted. The configuration of cluster explained in this article is basis for developing your own within your own additions or exclusions required by your own situation.

The issue: to develop fault tolerant cluster architecture for Distributed Erlang Application that will ensure the 24/7 service availability when the budged restricted.

First of all need to make clear the definition "distributed application". The main question - Why distributed? The reply on it might be looking like:

- Do you need high level of reliability and fault tolerance? - Yes, then you need to add redundancy to your solution, when falling of one element of cluster out will not affect your service.
- Is your service going to handle high-load? - Yes, then you have to have ability to scale your service.
- Is your cluster going to handle big-data storage? - Yes, then you need to have automated solution for scaling storage
- etc

This list might be prolonged further. All of this points describe the situation that your service isn't OK for one server solution an might be scaled in future. If your project is OK for one server - then this article not for your case.



This schema describing then solution that might be used by company or organisation of any size (trade company, hospital, education, small startup and etc) that is going to build own environment in-site or using data center for collocating

servers.

Efficiency of this solution has been proving by using it in:

- HTTP REST API for the case of using in mobile, desktop and web applications that might be OK for handling +30K connections per second (this value directly depend on kind of request the your application handling, it might be more or less)
- The small file storage within ability to be scaled in case of necessity
- Security Events Aggregator
- Application Usage statistic collector from mobile, desktop and web applications

The technology stack manifest is:

- **FreeBSD** (www.freebsd.org): the basis OS, the reasons for choosing this OS like basis explained in article "Why Freebsd?"
- **PfSense** (www.pfsense.org): this Open Source solution based on FreeBSD and provide high level of security for any company. The author of this article hasn't been using reinstalled devices delivered by this company, there was always using OS PfSense that is installed on standalone server equipped by few network interfaces. In most cases it very good and cheap solution within huge functionality for the case of exchanging CISCO devices.
- **FreeBSD Jails** (www.freebsd.org/doc/handbook/jails.html): the technology that is allowing to get kind of virtualisation at minimal cost and developers qualification. From begin this tech been developing for improving the level of security. In Linux-family presented the same kind of application, but practical usage shown the lower performance and higher resource requirements. FreeBSd Jail might be set up and being looking from the user and system administrator sides like standalone server. You might to instal on one physical server what count you want of Jails and every jail will bee looking like standalone server. If you will organise automation tools based on group techs Jenkins (jenkins.io) + BASH Script (Wiki Bash) + Ansible (www.ansible.com) you will be able to get the same level of automation that presented in Amason Web Service. It's not so difficult how it might looks from the first glance.
- **Ezjail** (Freebsd: Managing Jails with ezjail): the service application that is making jails handling procedures simpler. In case of your own necessity there are nothing difficult to write your own BASH Scripts that will help you manage Jails without outside applications.

All of this 4 points described all that you need from the technical stack for organising your own cluster. The architectural solution explained already has the scalable mechanism inside:

- every Jail looking like standalone server and you might to replicate any part of cluster on standalone physical server
- every DB Engine located on Jail, there for if you got huge load you might to move it on standalone server
- etc

The basis of it - UNDERSTANDING THAT YOU GOING TO GROW UP AND THEREFORE YOU INCLUDING SCALING MECHANISM FROM BEGIN. It's not so expensive and not so difficult even for small startup. It's only your own position towards the developing process.

This article hasn't been explaining all of details of this project realisation, there are huge list. It was only idea explanation. If you have questions - contact the author Alexandr Kirilov.

Follow author updates on **Linkedin**

Follow AR|BO|RE|US updates on **Twitter** and **Linkedin**