- Monolitic and why?
- faster develop
- its only my machine
- If the app is big. My machine is suffering
- scale vertical (best performance <-> best machine)
- 12 - factors (heroku)
- https://12factor.net/ I. Código base (Codebase) Un código base sobre el que hacer el control de versiones y multiples despliegues

II. Dependencias Declarar y aislar explícitamente las dependencias
III. Configuraciones Guardar la configuración en el entorno
IV. Backing services Tratar a los "backing services" como recursos conectables V. Construir, desplegar, ejecutar Separar completamente la etapa de construcción de la etapa de ejecución
V. Procesos Ejecutar la aplicación como uno o más procesos sin estado
VI. Asignación de puertos Publicar servicios mediante asignación de puertos
VII. Concurrencia Escalar mediante el modelo de procesos
VIII. Desechabilidad Hacer el sistema más robusto intentando conseguir inicios rápidos y finalizaciones seguras X. Paridad en desarrollo y producción Mantener desarrollo, preproducción y producción tan parecidos como sea posible
IX. Historiales Tratar los historiales como una transmisión de eventos
X. Administración de procesos Ejecutar las tareas de gestión/administración como procesos que solo se ejecutan una vez

- Lo que ya tenía I. Código base (Codebase)

II. Dependencias V. Construir, desplegar, ejecutar
III. Administración de procesos

- Dividir los servicios más pesados (las cosas clasicas, base de datos, autentaticacion) No son nuestra responsabilidad (IV. Backing services)
- Bases de Datos
- Sistemas de autenticación
- Control de versiones
- almacenamiento (Minio.io)
- Dividir las tareas en micro servicios (Como levantar todo a la vez)
- Desechar es facil !!!!!!
- Orquestar como se levanta la aplicación
- La ventaja de que los servicios pueden escalar independientemente

- (III. Configuraciones, VI. Procesos, IX. Desechabilidad)
- Production - Stage
- Tener dos ambientes separados para poder probar nuestro código y como funciona (stage no es nuestra pc) (X. Paridad en desarrollo y producción)
- Continuos Delivery and Deployment (test correctitud, mejor calidad de software)
- Unitest are important
- Code Coverage
- El deploy de la aplicación no debe requerir intervención humana
- Jenkis - Gitlab
- Lo más cercano que tenemos es un sistema altamente virtualizado
- Necesitamos mejores gestores de nube e incluir orquestadores
- Utilizar soluciones opensource como rancher OS para manejar de forma más sencilla Kubernetes
- Almacenamiento, BD, Servicios en Kubernetes (load balace, health check, etc, load balance, etc muchos problemas resueltos, logs)
- Hot refresh
- Escribir kubernetes is hard
- Plataform as a service (flynn.io)

Extra Notes:

- Buscar estructuras de software más horizontales (si alguien tiene una idea y la programó, no debe tardar más de un día en ponerla en producción)
- Ofrecer como servicio las mayores necesidad de los programadores (bases de datos, servicios de integración, repositorios) esto acelera la producción del software
- Los mejores patrones de software para estas arquitecturas