Arquitectura Proyecto Nuevo Turnero Etapa 2

Proyecto: Nuevo Turnero Etapa 2

Revisiones

Revisión	Autor	Fecha	Descripción
1.0	Mayco Fracarolli	30-09-2020	Versión Inicial
2.0	Julián Gigena	05-03-2021	Revisión

Índice

Propósito

En este documento se especifica la arquitectura de la aplicación web turnero. En la misma se detalla arquitectura a nivel de infraestructura, seguridad y desarrollo de la aplicación.

Antecedentes y justificaciones

Turnero es una aplicación que permite a los ciudadanos obtener turnos para realizar distintos trámites dentro de las dependencias del gobierno provincial, posibilitando el uso eficiente de tiempo y recursos de usuarios y de la gestión provincial y proveyendo orden en la atención.

Problemas en el diseño y despliegue de alguno de sus componentes comprometen la estabilidad del mismo, requiriendo un soporte manual permanente y el tiempo de indisponibilidad del mismo ha aumentado considerablemente.

Este deterioro en la disponibilidad ha sido tal que ha puesto en riesgo la reputación de la misma al poner en duda la utilidad de la herramienta para cumplir con el propósito para el cual ha sido diseñada.

Por tanto, su no disponibilidad no solo diluye los beneficios de la aplicación, sino que contribuye al desorden en la atención y al descontento de los usuarios durante este proceso.

Objetivo general

1. *Fase de estabilización:* proveer una mayor estabilidad a la aplicación para que cumpla con su rol específico. Esto incluye, confiabilidad y disponibilidad de la app. Esta fase prevé un proceso de reingeniería que le permita a la aplicación ser escalada más armónicamente.

2. *Fase de ingeniería:* Proveer mecanismos que consoliden los procesos de gestión de ingeniería que mejoren la mantenibilidad de la aplicación y mejore los procesos de despliegue.

3. *Fase de desarrollo:* Incorporar nuevos requerimientos de características que no han sido parte de la versión actual.

Aplicación Turnero: Componentes del servicio

Turnero se construye como aplicación web multicapa que sigue un patrón de diseño SPA (Aplicación de página única). A continuación, se describen cada uno de los componentes que componen la solución.

CiDi

Este componente está fuera del alcance de la solución, pero se representa aquí ya que es parte integral del ciclo de uso de la aplicación. Este componente es responsable de la autenticación de usuarios de aplicación. Una vez completado el proceso de autenticación, CiDi entrega una cookie identificando al usuario. Adicionalmente, CiDi provee una interfaz de servicio que permite validar las sesiones y obtener información del usuario (profile de usuario).

Spa Web App

El front-end se construye sobre AngularJS 1.6.4. Esta versión se encuentra en LTS (soporte de largo plazo) a partir del 1 de Julio de 2018 por el término de 3 años.

Este componente se divide en 3 módulos que corren como aplicaciones independientes, cada una destinada a un grupo de usuario particular. Este esquema tiene como propósito permitir escalar cada componente independientemente y poder desplegar cada componente de forma que se le pueda proveer restricciones de acceso que corresponda.

Módulo	Usuarios	Restricción de acceso	Autenticación[1]
Ciudadano	Ciudadanos y representantes de personas jurídicas	Abierto a internet	Cookie CiDi
Administrador	Agentes	Abierto a internet con certificado / Lista blanca	Cookie CiDi
Salón (Llamador y Recepción)	Usuario interno	Abierto a internet con certificado / Lista blanca	Autenticación interna basada en cookies

[1] ver apartado Seguridad Autenticación para más detalles

Para los módulos enlistados más arriba como dependientes de CiDi los usuarios son requeridos de autenticar en esa plataforma previo a poder acceder a la aplicación. La aplicación hace la redirección al portal de CiDi y su login externo como parte del proceso de autenticación.

Para los módulos enlistados no dependiendo de CiDi se utiliza **.net Identity** para el proceso de autenticación.

Las notificaciones asíncronas se realizan a través de un cliente signalR.js. En caso de fallo en el mecanismo principal de notificaciones, se implementa un mecanismo backup de pulling.

Web API

Este componente es responsable de exponer servicios web que serán consumidos desde el front-end y desde el componente de hub. Basado en .net core 2.2 WebApi, C# 7.3.

Este componente es el responsable de la conexión con la base de datos a través del cliente Oracle.

El catálogo de métodos de servicio expuesto se publica a través de swagger 2.0. Sin embargo, esta publicación solo estará disponible si se consulta desde la red interna de gobierno.

Hub

Este componente es responsable de concentrar y transmitir todas las notificaciones asíncronas a los clientes registrados. Expone servicios web que serán consumidos desde el front-end / back-end. Basado en .net core 2.2 WebApi, C# 7.3. Usa signalR Hubs para la comunicación basada en WebSockets, Server-Sent Events o Long Polling según corresponda o soporte el cliente.

Caché

Este componente es responsable para el almacenado y propagación de notificaciones asíncronas y resultados de consultas para mejora de tiempos de respuestas. Basado en Redis 5.0.4.

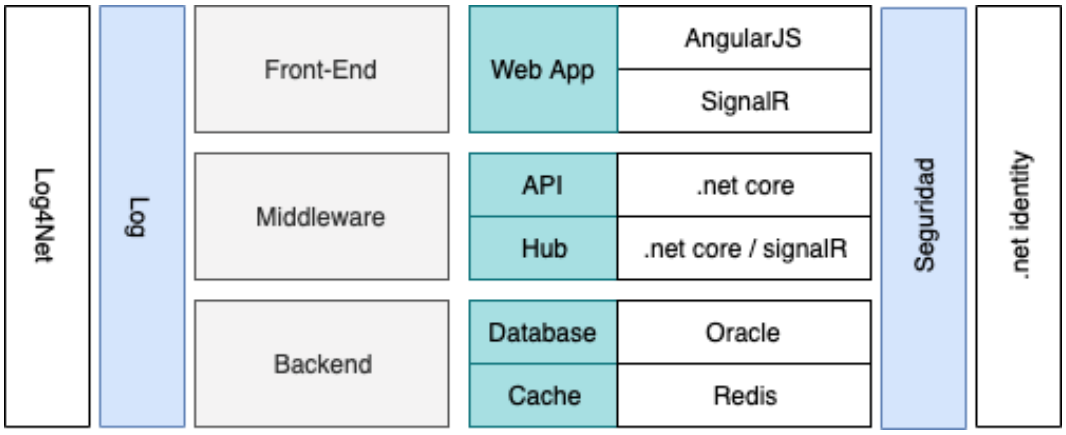
Base de datos

Como parte de la versión original de la solución, parte de la lógica de negocio de la misma reside en la base de datos en forma de paquetes. Este componente se mantiene sin modificación en su alcance. La base de datos a utilizar son las organizacionales.

Ambiente	Instancia	Esquemas
Producción	CBA1	TURNOS GUIAS_TRAMITES T_COMUNES DOM_MANAGER RCIVIL GESTION_CIUDADANOS NUEVO_SUAC
Preproducción	CBA1PRE	
Testing	CBA1T	
Desarrollo	CBA1D	

Stack de tecnologías

Aplicación



Front-end

Angular JS: La aplicación se despliega como una aplicación de sola página utilizando AngularJS 1.6.4. El desarrollo original de la aplicación generó una dependencia con la versión 1.X de la librería y no se justifica la migración a Angular.io.

SignalR: es una librería para aplicaciones ASP.NET que simplifica el proceso de agregar funcionalidad basada en interacción web en tiempo real (capacidad de poder enviar contenido desde el servidor a determinadas aplicaciones clientes que lo tendrán disponible de forma inmediata, en vez de que el servidor tenga que

esperar que el cliente le solicite nueva información), permitiendo lograr un esquema de conexión bidireccional completo.

Middleware

.NET Core: Es una plataforma de desarrollo de uso general de código abierto de cuyo mantenimiento se encargan Microsoft y la comunidad .NET. Es multiplataforma, admite Windows, macOS y Linux y puede usarse para compilar aplicaciones de dispositivo, nube e IoT. La elección de este framework (en su versión 2.2) asegura una mejor performance actual y futura, una mayor flexibilidad en los mecanismos de despliegue y la elección del sistema operativo.

SignalR: Este componente, ya descrito en la sección front-end, se distribuye dentro del middleware actuando como data hub para la distribución de eventos entre los suscriptores en front-end.

Nginx: Nginx 1.17.x se utiliza para el servicio del contenido estático de la aplicación (Web Server). Este contenido no es más que los recursos de la aplicación angular (HTML, imágenes, estilos y archivos de JavaScript).

Back-end

Oracle: Los datos y esquemas de la aplicación se despliegan en base datos Oracle 12.x en las instancias de gobierno. Gran parte de la lógica de la aplicación se encuentra en la forma de procedimientos almacenados dentro de esta y es parte de un plan futuro de migración hacia el middleware para desacoplar la lógica de esta y darle flexibilidad y mantenibilidad reduciendo las dependencias de esta.

Redis: es un motor de base de datos en memoria, basado en el almacenamiento en tablas de hashes (clave/valor) pero que opcionalmente puede ser usada como una base de datos durable o persistente.

Transversales

Log de errores: Los mecanismos de monitoreo son descritos en un apartado diferente. El proceso de reporte de actividad se basa en la utilización de log4net para consolidación de los mismos de forma flexible en distintos medios. De forma básica los logs que produce la aplicación son colectados para ser centralizados en ELK.

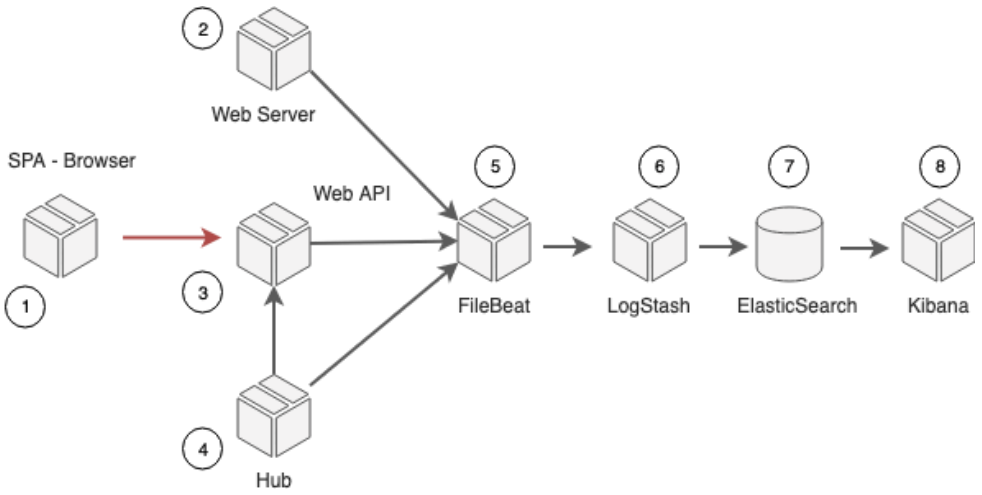
Seguridad: La seguridad de usuario de la aplicación se delega en CiDi. Internamente la aplicación mantiene una cookie de reconocimiento de sesión, basado en los mecanismos internos de .Net core.

Monitoreo Turnero – Componentes

Los componentes descritos a continuación representan la colección de componentes dispuestos para el monitoreo y control de la aplicación. Esta colección se divide en componentes de monitoreo por métricas y componentes de monitoreo para traza de eventos.

Estos stacks no se consideran críticos para la operación, y son solo soporte a esta. Por esta razón no son desplegados en esquemas de alta disponibilidad respetando la economía de recursos.

Traza de eventos



El stack de traza de eventos permite recolectar información log de los distintos componentes y consolidarlos en un solo punto. Esto permitirá usarlo principalmente para la investigación de incidentes y para la toma de acciones de soporte proactivo.

Dentro del gráfico anterior se distinguen componentes de la aplicación ya descritos, considerados fuente de eventos:

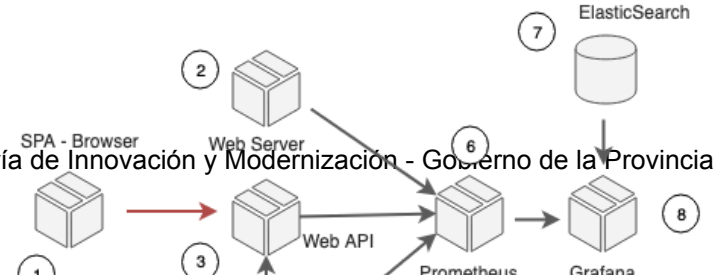
1. SPA - Browser
2. Web server
3. Web API
4. Hub

Por otro lado, los siguientes componentes hacen parte del trace stack:

5. FileBeat: Colecta archivos de logs y los envía a logstash para su formateo y filtrado de ser necesario.
6. LogStash: Filtrado y formateado de las diferentes fuentes de datos.
7. ElasticSearch: Herramienta de búsqueda y análisis altamente distribuida
8. Kibana: Provee capacidad de visualización sobre el contenido indexado por ElasticSearch.

Métricas

El stack de métricas permite recolectar métricas propias de los componentes de aplicación, de servicios de aplicación, y un limitado grupo de métricas de software de base (memoria y CPU).



Al igual que en el caso anterior los siguientes componentes son considerados fuente de métrica.

1. SPA - Browser
2. Web server
3. Web API
4. Hub
5. Redis
6. ElasticSearch

Notar que en este caso ElasticSearch es parte de la fuente de datos ya que la información colectada para analíticas es interpretada y presentada en forma métricas en este stack.

Los siguientes componentes son parte de:

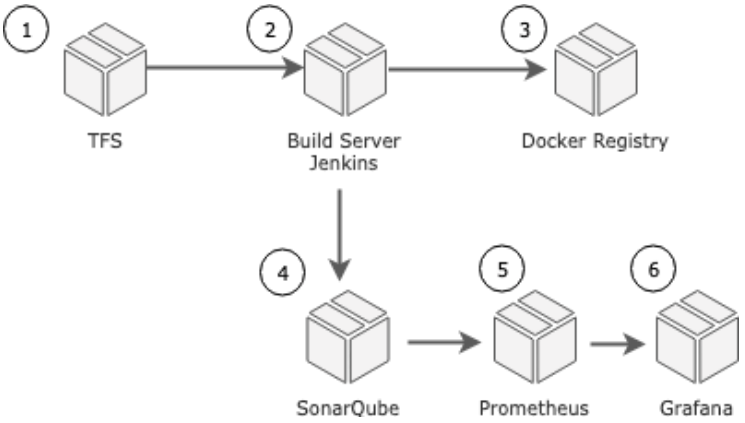
7. Prometheus: conjunto de herramientas de monitoreo y alerta de sistemas.
8. Grafana: Provee capacidad de visualización sobre las métricas consolidadas en Prometheus

Soporte a ciclo de vida de la aplicación

El stack de soporte a ALC (Application Life Cycle), colaboran con el manejo de la configuración, construcción, validación y despliegue de los componentes de la aplicación. Algunos de estos componentes son transversales a más de un proyecto (aunque por lo pronto utilizan instancias propias de turnero hasta que estén disponibles globalmente).

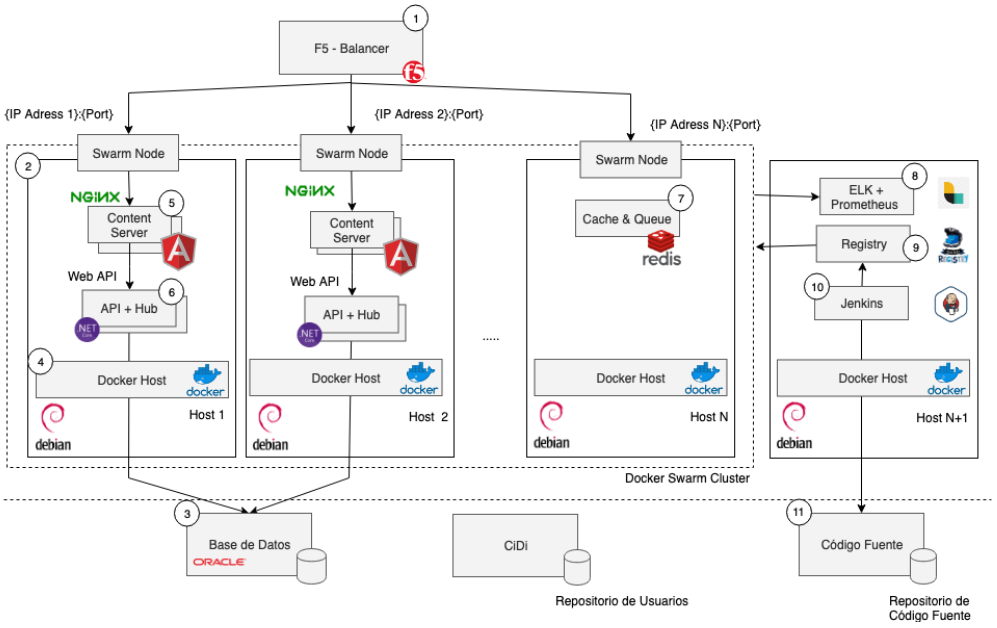
Como se verá, algunos de los componentes aquí descritos coinciden con los de monitoreo ya que el resultado de estos procesos debe consolidarse en estas herramientas.

1. TFS: CM tool que se utiliza para el manejo de la configuración del código fuente.
2. Jenkins: Servidor de Compilación y despliegue.
3. Private Docker Registry: Repositorio de imágenes de contenedores privados.
4. SonarQube: Validación de la calidad y métricas de código.
5. Prometheus: Repositorio de métricas que en este caso son en relación con la calidad del código.
6. Grafana: Visualización de métricas.



Infraestructura

Componentes de la infraestructura



El diagrama anterior muestra la infraestructura a utilizar y el despliegue de componentes identificados

1. F5 – Load balancer: Responsable de distribuir la carga entre los diferentes nodos del cluster.
2. Servidor virtualizado con una imagen Debian 9: Aloja el servicio de Docker en cada uno de los nodos o instancia individual.
3. Base de Datos Oracle: compartida por todas las aplicaciones del gobierno.
4. Nodo Docker: Responsable de dar soporte a los contenedores de las distintas aplicaciones.

5. Servidor de aplicaciones: responsable de servir el contenido de la aplicación web. Dado que esta es un SPA, todo el contenido de la misma es considerado estático.
6. Componente de API de la aplicación: Estos son dos componentes .net core web api que corren como stand-alone.
7. Caching de la aplicación para mantenimiento de respuestas de servicio y distribución de notificaciones.
8. Stack de monitoreo de dockers.
9. Docker Registry para el almacenamiento privado de imágenes de servicios.
10. Jenkins: Integración continua y generación de imágenes de servicios.
11. TFS. Repositorio de código fuente.

Ambientes

Todos los ambientes reproducen el mismo esquema de infraestructura y corren las mismas imágenes de contenedores, siendo la única diferencia la cantidad de nodos que integran el cluster.

Ambientes	**Número de Nodos (N)***	**Nodo de Soporte****
Desarrollo	2	1
Testing	2	1
Pre-Producción	2	1
Producción	8	1

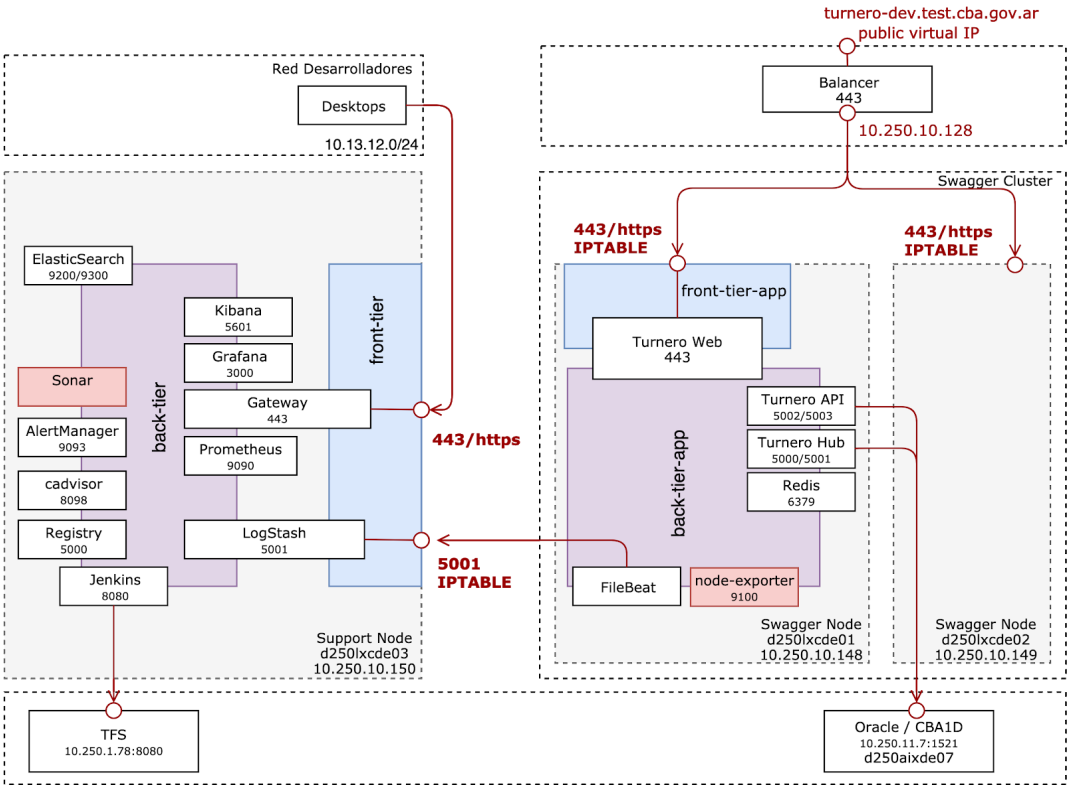
* *N: Número de nodos según el gráfico más arriba.*

** *Nodo de Soporte: Nodo en el que se despliegan herramientas de integración continua y monitoreo.*

Desarrollo

FQDN	IP / Name	Destino	Puertos expuestos
turnero-dev.test.cba.gov.ar (balanceado) monitor-turnero-dev.test.cba.gov.ar	d250lxcde01.gobiernocba.gov.ar (10.250.10.148)	Swarm Cluster Nodo 1	443/https - Internet 22/ssh - Interno
	d250lxcde02.gobiernocba.gov.ar (10.250.10.149)	Swarm Cluster Nodo 2	
registry-turnero-dev.test.cba.gov.ar build-turnero-dev.test.cba.gov.ar trace-turnero-dev.test.cba.gov.ar metrics-turnero-dev.test.cba.gov.ar	d250lxcde03.gobiernocba.gov.ar (10.250.10.150)	Nodo Soporte	

Diagrama de entornos



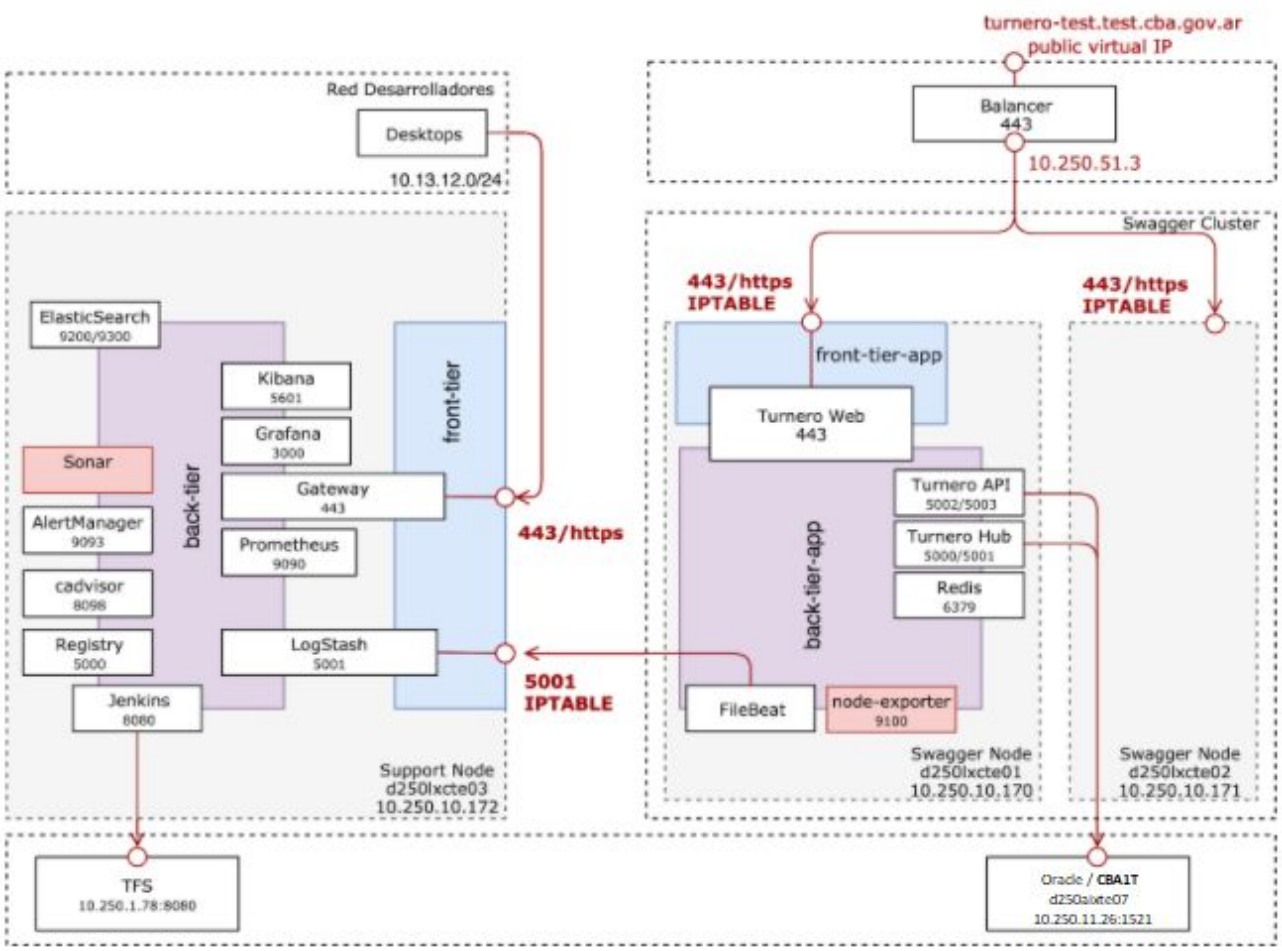
Reglas Firewall

Origen	Destino	Tipo	Acción

Testing

FQDN	IP / Name	Destino	Puertos expuesto
turnero-test.test.cba.gov.ar (balanceado) monitor-turnero-test.test.cba.gov.ar	d250lxcte01.gobiernocba.gov.ar (10.250.10.170)	Swarm Cluster Nodo 1	443/https - Internet 22/ssh - Interno
	d250lxcte02.gobiernocba.gov.ar (10.250.10.171)	Swarm Cluster Nodo 2	
registry-turnero-test.test.cba.gov.ar build-turnero-test.test.cba.gov.ar trace-turnero-test.test.cba.gov.ar metrics-turnero-test.test.cba.gov.ar	d250lxcte03.gobiernocba.gov.ar (10.250.10.172)	Nodo Soporte	

Diagrama de entornos



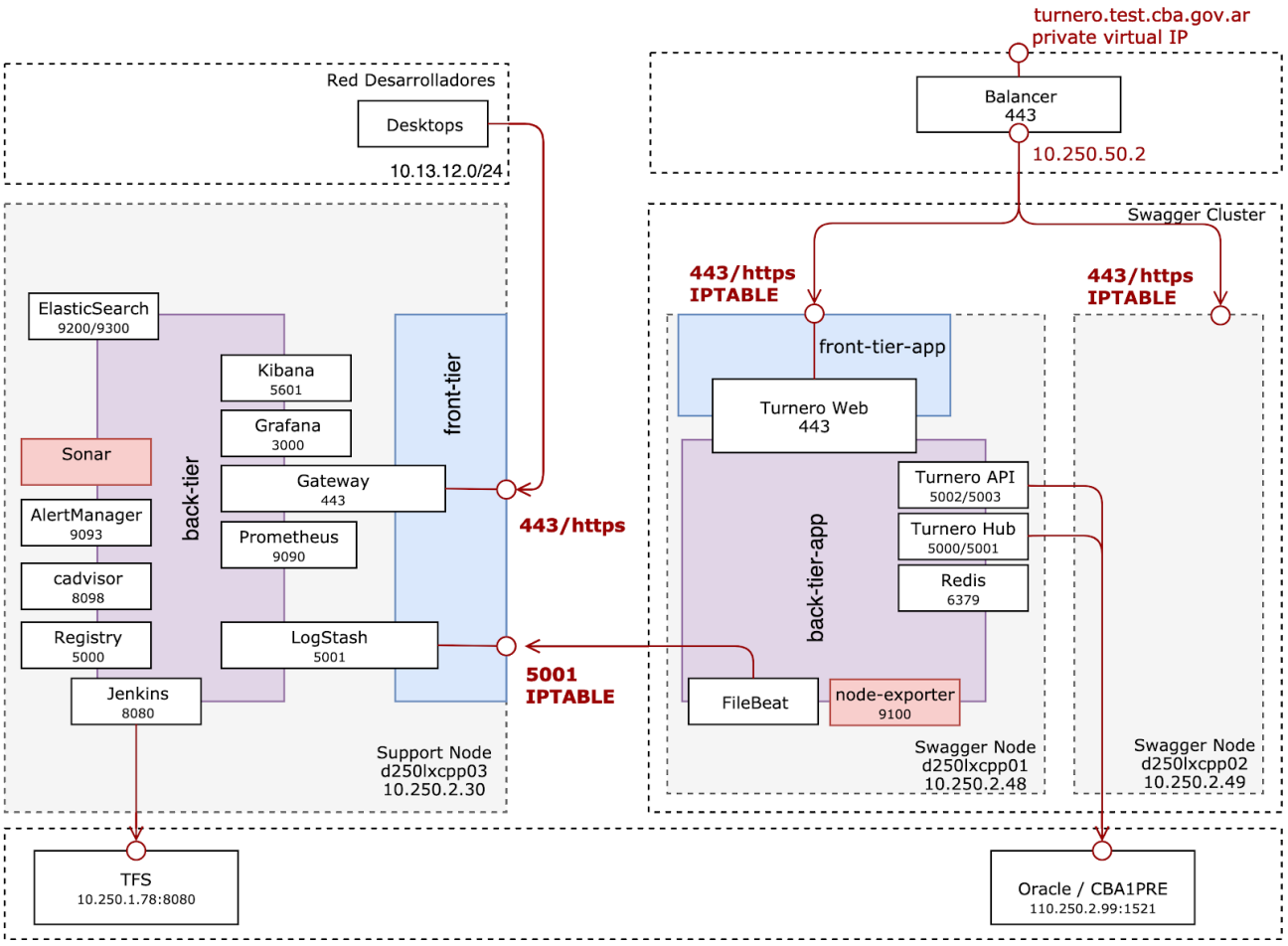
Reglas Firewall

Origen	Destino	Tipo	Acción

Pre-Producción

FQDN	IP / Name	Destino	Puertos expuesto
turnero.test.cba.gov.ar (balanceado) monitor-turnero.test.cba.gov.ar	d250lxcpp01.gobiernocba.gov.ar (10.250.2.48)	Swarm Cluster Nodo 1	443/https - Internet 22/ssh - Interno
	d250lxcpp02.gobiernocba.gov.ar (10.250.2.49)	Swarm Cluster Nodo 2	
registry-turnero.test.cba.gov.ar build-turnero.test.cba.gov.ar trace-turnero.test.cba.gov.ar metrics-turnero.test.cba.gov.ar	d250lxcpp03.gobiernocba.gov.ar (10.250.2.30)	Nodo Soporte	

Diagrama de entornos



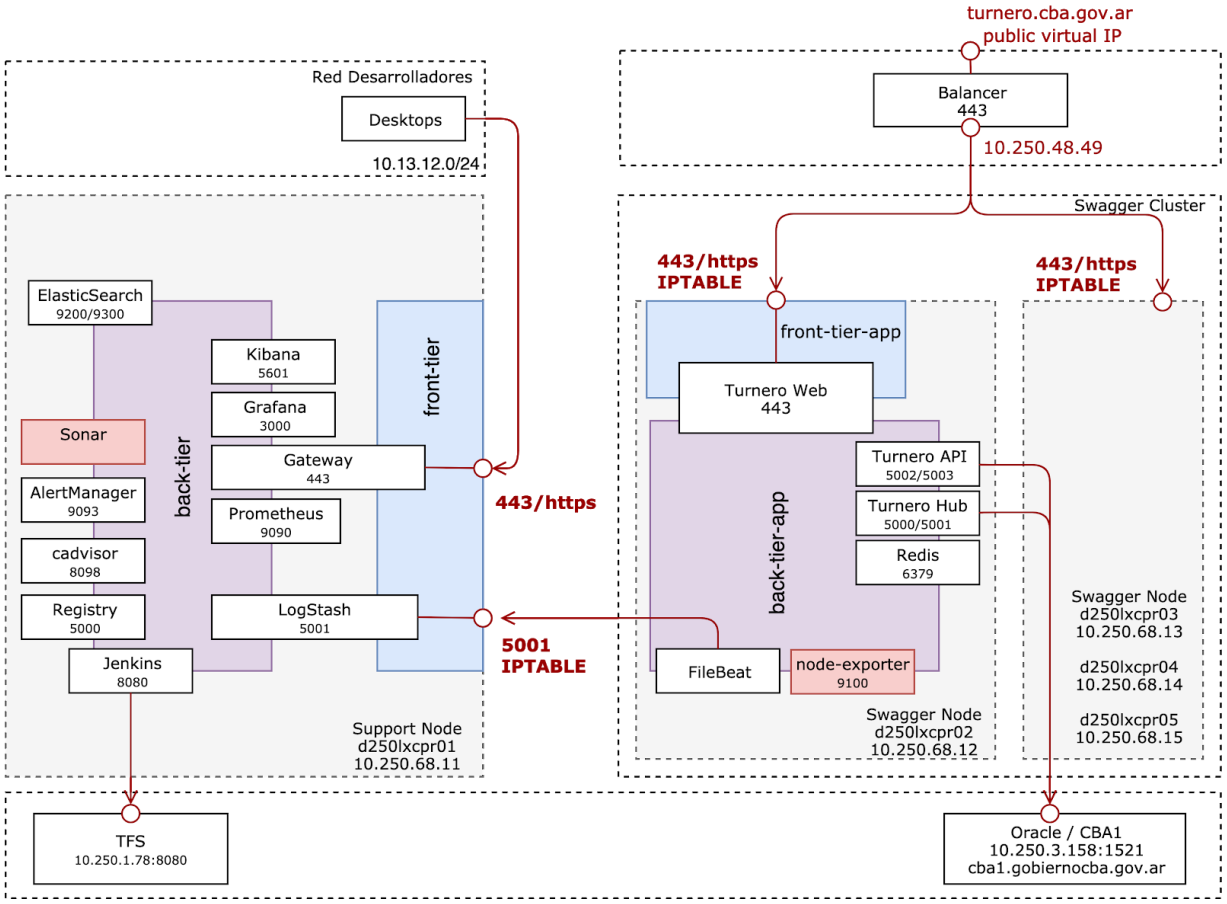
Reglas Firewall

Origen	Destino	Tipo	Acción

Producción

FQDN	IP / Name	Destino	Puertos expuesto
turnero.cba.gov .ar (balanceado) monitor-turnero.cba.gov.ar	d250lxcpr02.gobiernocba.gov .ar (10.250.68.12)	Swarm Cluster Nodo 1	443/https - Internet 22/ssh - Interno
	d250lxcpr03.gobiernocba.gov .ar (10.250.68.13)	Swarm Cluster Nodo 2	
	d250lxcpr04.gobiernocba.gov .ar (10.250.68.14)	Swarm Cluster Nodo 3	
	d250lxcpr05.gobiernocba.gov .ar (10.250.68.15)	Swarm Cluster Nodo 4	
	d250lxcpr26.gobiernocba.gov .ar (10.250.68.47)	Swarm Cluster Nodo 5	
	d250lxcpr27.gobiernocba.gov .ar (10.250.68.48)	Swarm Cluster Nodo 6	
	d250lxcpr28.gobiernocba.gov .ar (10.250.68.49)	Swarm Cluster Nodo 7	
	d250lxcpr29.gobiernocba.gov .ar (10.250.68.50)	Swarm Cluster Nodo 8	
registry-turnero.test.cba.gov .ar build-turnero.test.cba.gov.ar trace-turnero.test.cba.gov.a r metrics-turnero.test.cba.gov .ar	d250lxcpr01.gobiernocba.gov .ar (10.250.68.11)	Nodo Soporte	

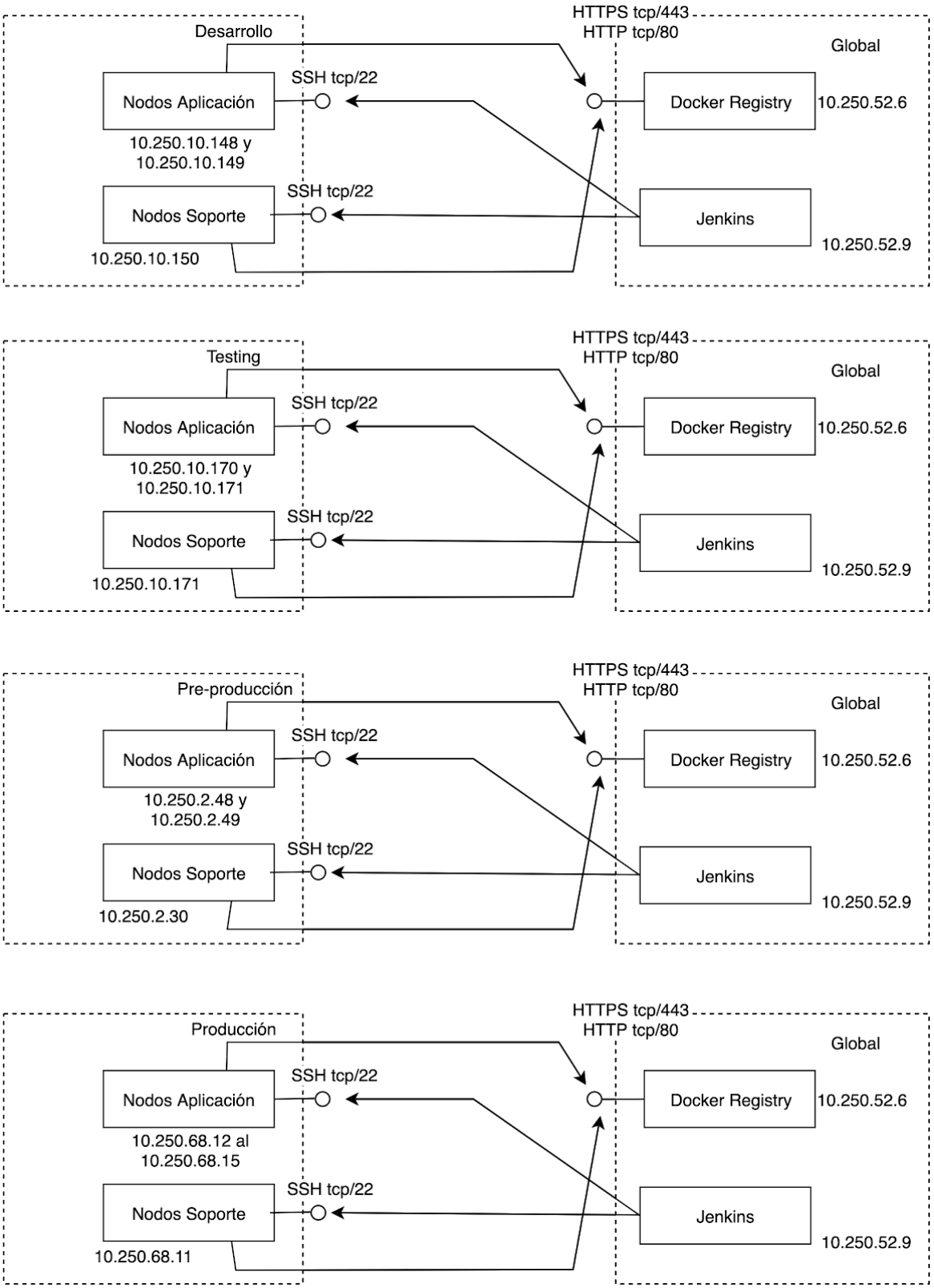
Diagrama de entornos

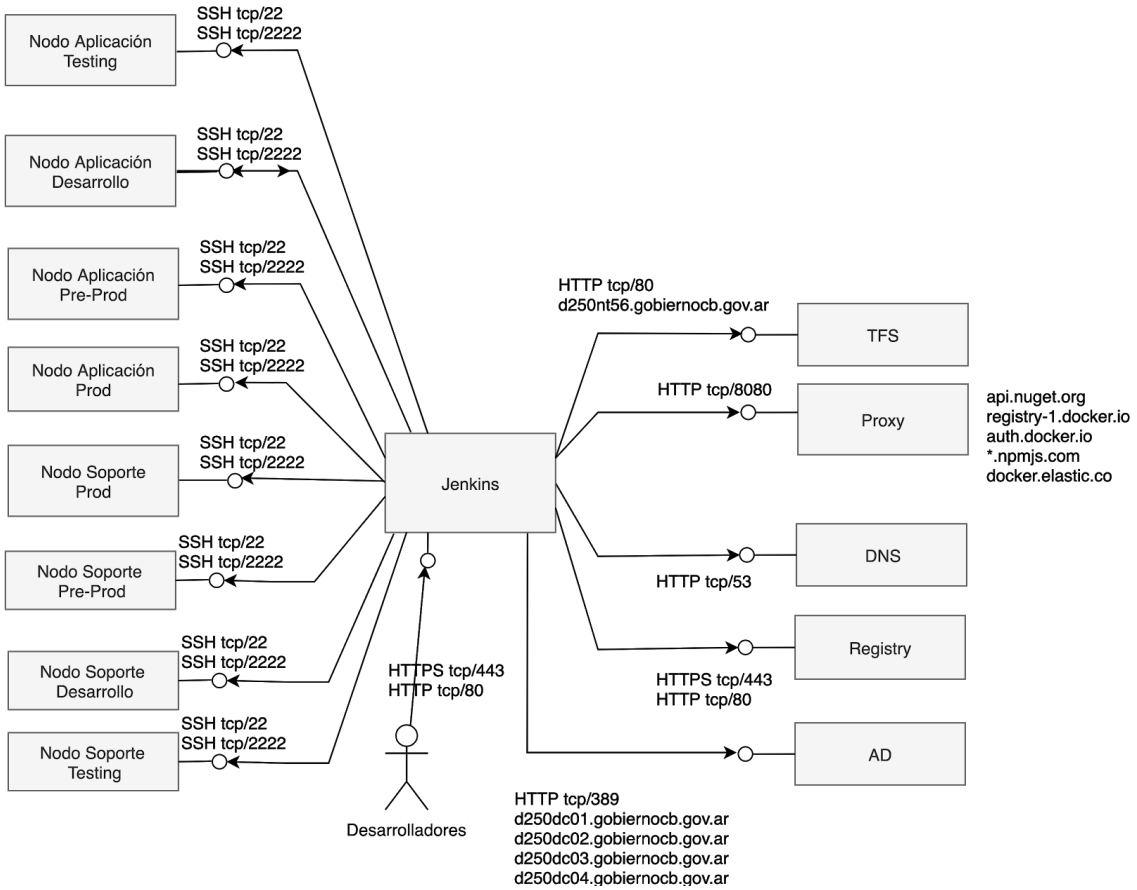


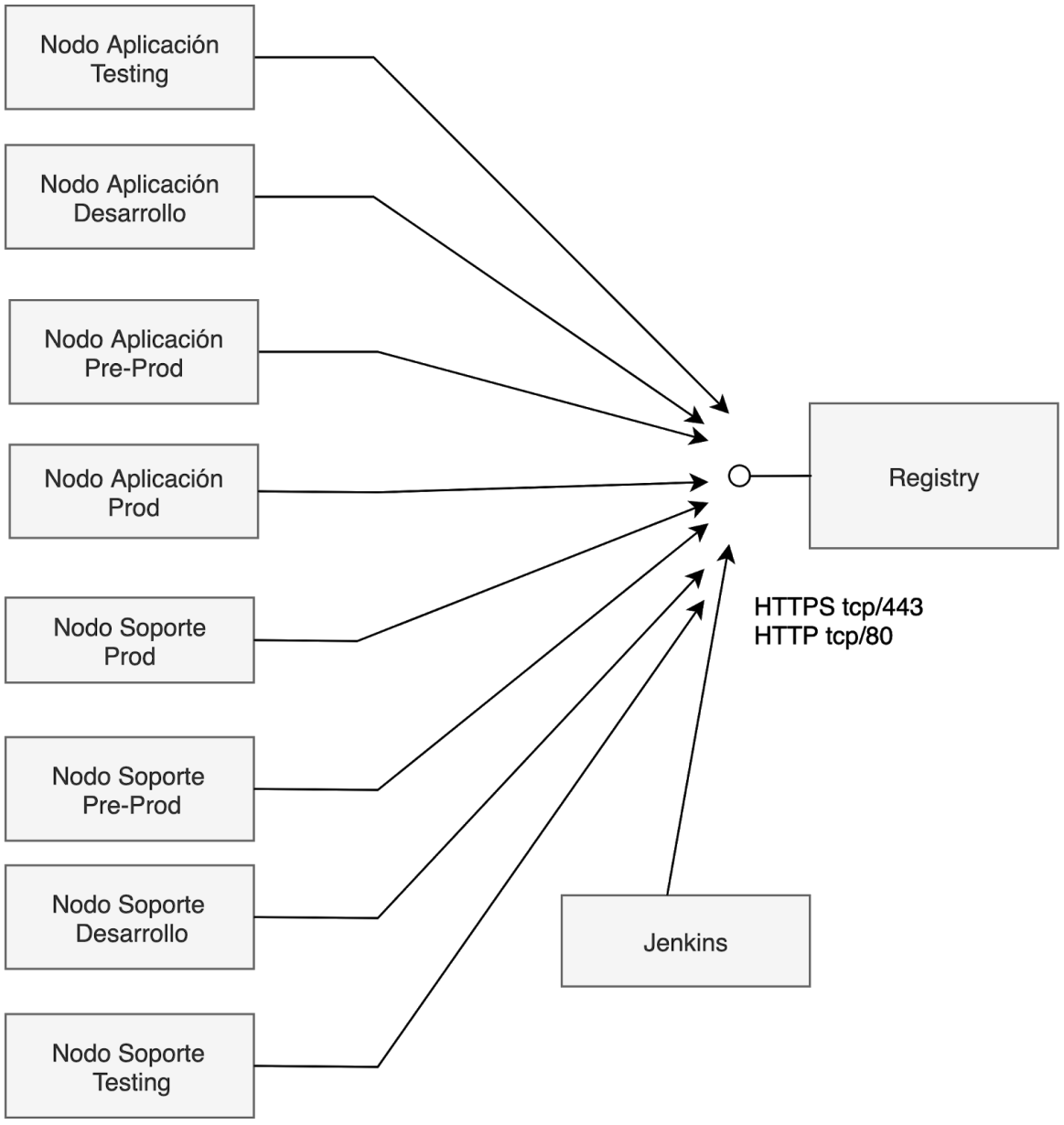
Reglas Firewall

Origen	Destino	Tipo	Acción

Esquema de conexiones Docker Registry/Jenkins globales




Nodo Aplicación Testing
Nodo Aplicación Desarrollo
Nodo Aplicación Pre-Prod
Nodo Aplicación Prod
Nodo Soporte Prod
Nodo Soporte Pre-Prod
Nodo Soporte Desarrollo
Nodo Soporte Testing
SSH tcp/22
SSH tcp/2222
Jenkins
Desarrolladores
HTTPS tcp/443
HTTP tcp/80
HTTP tcp/80
d250nt56.gobiernocb.gov.ar
TFS
HTTP tcp/8080
Proxy
api.nuget.org
registry-1.docker.io
auth.docker.io
*.npmjs.com
docker.elastic.co
DNS
HTTP tcp/53
Registry
HTTPS tcp/443
HTTP tcp/80
AD
HTTP tcp/389
d250dc01.gobiernocb.gov.ar
d250dc02.gobiernocb.gov.ar
d250dc03.gobiernocb.gov.ar
d250dc04.gobiernocb.gov.ar


Nodo Aplicación Testing
Nodo Aplicación Desarrollo
Nodo Aplicación Pre-Prod
Nodo Aplicación Prod
Nodo Soporte Prod
Nodo Soporte Pre-Prod
Nodo Soporte Desarrollo
Nodo Soporte Testing
Registry
HTTPS tcp/443
HTTP tcp/80
Jenkins

Componentes de Soporte al Servicio

#	Componente	Responsabilidad	Comentarios
1	ELK + Prometheus/Grafana	Monitoreo	Unificación de logs de aplicación y soporte. Publicación de métricas de aplicación.
2	Registry	Construcción y Despliegue	Registro para el almacenamiento de imágenes.
3	Jenkins / Sonar	Construcción y Despliegue	Herramienta para la construcción, prueba despliegue automática
4	TFS	Construcción y Despliegue	Repositorio de Código fuente. Manejo de la configuración.
5	Portainer	Administración del Cluster	Portal para la administración del cluster swarm
6	Gateway	Punto de acceso a aplicaciones	Punto de entrada para las aplicaciones Kibana, Docker Registry, Grafana, Jenkins

Seguridad

Autenticación y Autorización

El proceso de autenticación a utilizar depende del módulo al que se esté accediendo:

1. Para módulos que requieren acceso a usuarios de personas físicas el proceso de autenticación se realiza a través de CIDI como proveedor de identidades (Identity Provider). Este portal devuelve una cookie y una interfaz de servicio que puede ser usada para validar la misma. Esta es usada a su vez dentro de la aplicación para mantener la identidad y obtener el perfil de la persona autenticada (nombre, apellido, correo, etc).
2. Para módulos que no cuentan con la posibilidad de la autenticación de personas físicas, la autenticación se realiza por validación de tokens. Este es el caso de los tótems llamadores en donde no hay un usuario físico detrás del mismo.

El proceso de autorización se compone de dos niveles.

1. A nivel de servicio (backend):
    a. Cada controlador de servicio requiere una autenticación válida para poder llamar a cualquier método.
    b. Cada método dentro del servicio identifica si ese método está disponible para todos los roles o para roles específicos (ciudadanos, administrativos, totems).
    c. Cada método puede eventualmente pedir un nivel de acceso adicional de rol otorgando un segundo nivel de granularidad (por ejemplo acceso a leer agendas).
    d. Los métodos que operan sobre información específica a nivel de usuario, restringen la visibilidad del contenido a nivel de propiedad del usuario. Por ejemplo el método traer calendario implica traer el calendario de usuario autenticado, no pudiendo traer cualquier calendario (man in the middle, service spoofing, impersonalización).

2. A nivel de la aplicación (front-end). La aplicación habilita o no visualmente la disposición de opciones en pantalla según la persona y el rol de la misma.

Permisos de Usuario de SO

Los siguientes usuarios deben estar disponibles a nivel de sistema operativo en cada uno de los nodos que componen la solución.

Usuario	Permisos	Propósito
devuser	debe pertenecer al grupo de sudoers y docker	Usuario para el proceso de despliegue en este procedimiento.
srv_build	acceso a /opt/turnero, grupo docker, ssh passwordless rsa.	Usuario de servicio del proceso de Integración continua para la conexión ssh entre Jenkins Master y Jenkins Agente.
svc_turnero_TFS	Acceso a TFS para el proyecto turnero	Usuario de servicio que permite extraer el código del repositorio TFS dentro del proceso de compilación.
ctr_jenkins (10101)	Usuario regular	Usuario proxy para mapear usuarios de contenedor y host en al volumen
sa_turnero (10102)	Usuario regular	Usuario proxy para mapear usuario de contenedor y host en los volúmenes de componentes de proyecto.

Acceso Servicios

Accesibilidad Externa

Los siguientes accesos representan la accesibilidad a la aplicación desde fuera de la infraestructura del datacenter deben estar disponibles independientemente del ambiente.

Servicio	Puerto	Origen	Adicional[2]
Turnero Ciudadano	443 HTTPS	*	-
Turnero Administrativo	443 HTTPS	*	-
Turnero Llamador	443 HTTPS	*	Certificado cliente (pendiente de definir)

[2] Mecanismos de Seguridad Adicionales: La distribución física de los distintos tipos clientes y la forma de acceso a las aplicaciones obliga a que las aplicaciones sean expuestas de forma abierta a internet. Sin embargo es conveniente que cada módulo de la aplicación cuente con un nivel de accesibilidad distintos

Accesibilidad Interna

Los siguientes accesos representan la accesibilidad entre containers o desde dentro de la red de gobierno (según se indica en la columna origen)

Componente	Puerto	Origen	Comentarios
API	HTTPS:5001/tcp	Inter docker	
Hub	HTTPS:5003/tcp	Inter docker	
Pasarela	HTTPS:443/tcp	Inter docker	Gateway al resto de los componentes
Swarm	HTTPS:443/tcp	F5 Balancer	Cada nodo restringen 443 para que sean accesibles sólo desde F5.
Swarm	2376/tcp 2377/tcp 7946/tcp 7946/udp 4789/udp	Intra nodos hosts	Acceso de swarm de cluster.
Host Debian	SSH:22	Gobierno	Solo para entornos no productivos

TBC: Requiere ser refinado.

Desarrollo

Herramientas para el desarrollo

- Visual Studio 2017
- PL/SQL Developer 8
- Erwin Data Modeler 7

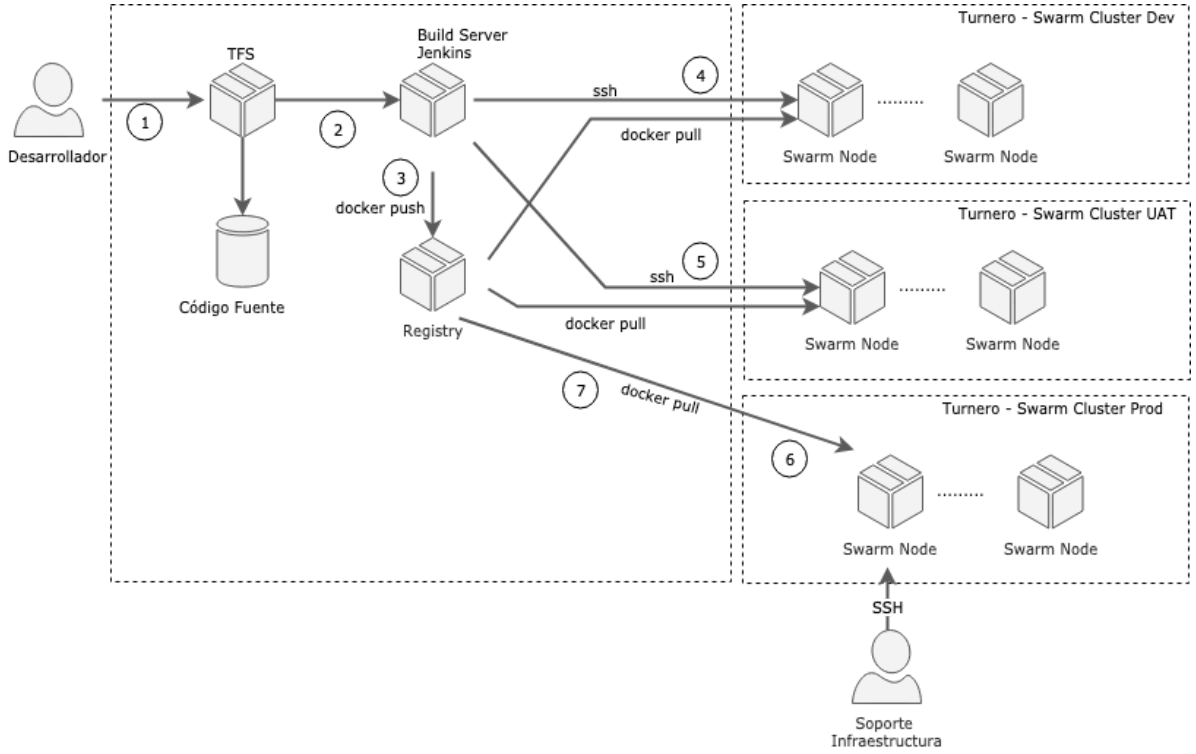
- Control de versiones de código fuente, Team Foundation Server 2015
- Gestión de proyectos, Team Foundation Server 2015.
- Metodología de trabajo utilizada: AGILE (SCRUM)
- Pruebas de performance y stress: JMeter 5.1.1, Postman Collection Runner

Validación de la Calidad del Entregable

(Quality Gates / Sonar)

Procedimientos de Construcción, Empaquetado y Despliegue

El proceso de construcción, empaquetado y despliegue automatizado se describe a continuación. Este procedimiento tendrá lugar finalmente cuando Jenkins y Docker Private Registry sean desplegados transversalmente en la infraestructura de gobierno.



Este proceso cambia con la incorporación de componentes globales de docker registry y jenkins.

Secuencia de Despliegue de Desarrollo

1. El desarrollador completa un push de código a TFS.
2. TFS envía una notificación a Jenkins de que un evento se ha producido a través de WebHooks configurados en TFS. Para esto es necesario que TFS pueda alcanzar Jenkins en el puerto 443.
3. Jenkins toma una copia del código y ejecuta su secuencia de compilación que incluye, descarga de paquetes, compilación y ejecución de test cases. Si ningún error se produce en este proceso, Jenkins genera las imágenes de docker y ejecuta un push al registry definido en la configuración de la receta de compilación (Parámetro de compilación DOCKER_REGISTRY). Las imágenes construidas se adjuntan al tag de latest.
4. Inicia el proceso de despliegue en entorno de desarrollo. Este se realiza automáticamente vía SSH contra el nodo principal del cluster docker. Para que esta conexión sea posible debe

generarse una configuración passwordless entre el servidor donde corre Jenkins y el servidor donde corre el nodo principal de docker, utilizando un usuario específico de despliegue, con solo pertenencia al grupo docker (sudoer no requerido y no debe tenerlo). Este usuario solo debe tener acceso a la carpeta /opt/turnero.
Jenkins ejecuta un script ansible para realizar las siguientes tareas:

- Copiar docker-compose.yml al directorio /opt/turnero
- Ejecutará un login al registry donde se encuentran las imágenes previamente construidas.
- Ejecutar el redespliegue de stack de swarm. En este despliegue se toman las imágenes marcadas como latest.

Secuencia de Despliegue de UAT

El proceso en el caso de UAT transcurre de la misma forma que en el caso de desarrollo, excepto la versión de código a compilar y desplegar debe incluir un Label en TFS indicando el RC (Release candidate) que identifica al freeze como requisito previo. La convención de nombres para las versiones es *major.minor[.maintenance]-RC* (por ejemplo *3.0.0-RC*). En este caso el parámetro de Label a extraer debe ser *L3.0.0*.

Este despliegue se inicia automáticamente pero, a diferencia del despliegue en desarrollo, este despliegue requiere especificar el parámetro de compilación en Jenkins para especificar la versión que se quiere desplegar.

El minor number (o el maintenance) se debe actualizar en cada despliegue.

Secuencia de despliegue de Producción

El proceso en el caso de Producción debe desplegar la imagen generada en UAT y promoverla como imagen de release. Para esto en el parámetro de versión de compilación se debe especificar la versión a promover. La convención de nombres es *major.minor[.maintenance]* (por ejemplo *3.0.0*).

Se tomará la imagen correspondiente a la versión (si es *L3.0.0* se tomará la imagen docker *3.0.0-RC*) desde el registry y se le aplicará el label de release correspondiente si se hará push al registry nuevamente.

Durante el proceso el docker swarm se actualiza con ese número de versión.

El minor number (o el maintenance) se debe actualizar en cada despliegue.

A diferencia del despliegue en UAT este despliegue se realiza manualmente y debe especificarse la imagen a promover/desplegar en el parámetro de versión.

Nota Importante: Este proceso no está en uso aún dado las limitaciones antes expresadas y no se ha creado el Jenkinsfile correspondiente al entorno de producción. El proceso que se describe a continuación es el mecanismo alternativo provisorio.

Secuencia de Despliegue de Producción Provisoria

Dado que no existe una instalación de docker registry común a todos los entornos, no es posible migrar imágenes entre estos como se especifica en el esquema anterior. Por lo tanto el siguiente esquema surge como solución provisoria para suplir esa falta.

El proceso en el caso de Producción transcurre de la misma forma que en el caso de UAT excepto que se debe generar un label en TFS identificado con la versión que se está empaquetando. La convención de nombres es *major.minor[.maintenance]* (por ejemplo 3.0.0). Este label en este esquema provisorio debe coincidir con el label de RC generado en UAT para mantener las versiones.

A diferencia del despliegue en UAT este despliegue se realiza manualmente en donde se especifica la versión a desplegar utilizando el parámetro de compilación de versión.

Nota Importante: Este procedimiento (a validar), no está inicialmente disponible. Mientras tanto Jenkins y Registry por ambientes serán dispuestos ya que el único componente que tendrá acceso desde todos los ambientes por lo pronto es TFS.