CT:

1. Desde el 2010 – 11 junto con el manifiesto ágil
2. La base es la automatización
3. El área es DevOps
4. Pruebas temprano, frecuente en “todas partes”

CPT Con Azure:

1. Configuración: Se configuran los criterios de aceptación de performance y se habilita Azure DevOps como repositorio de TestWare.
2. Ejecución: Se realiza la ejecución de las pruebas performance. Aunque actualmente es manual, el objetivo es que sea automática e integrada al desplegarse los componentes.
3. Visualización en línea de la ejecución: Se tiene acceso en línea a los resultados de cada prueba, generando comparativos históricos.
4. Resultados: Los resultados quedan registrado en Azure, dando continuidad al ciclo y se generan los reportes de manera automática.

Características:

1. Incrementar el time and date to market
2. Incrementar la cobertura de pruebas
3. Incrementar la autonomía de pruebas
4. Incrementar el feddback temprano
5. Aporta el cambio del enfoque hacia el “Testing Continuo”
6. Colaboración entre equipos
7. Pruebas inmersas de DevOps (Integración Continua)
8. Habilitar el pipeline exclusivo para Performance
9. Permite el reúso de componentes
10. Efectividad y frecuencia de pruebas
11. Validaciones automáticas
12. Validaciones automáticas de criterios de aceptación y Quality Gates de Performance
13. Generación de métricas e indicadores

Beneficios:

1. Administración centralizada
2. Fácil de incrementar y reutilizar
3. Validación automática de los criterios de aceptación
4. Incrementar la velocidad de entrega
5. Genera visibilidad en las tendencias e informes

Prácticas CPT:

1. Archivos principales:
   1. Build.jmx
   2. Script de JMeter
   3. BUG
   4. Tokens
   5. Carpeta de Data, almacenando .csv
   6. Archivos .yml (Características que tienen la ejecución)
   7. JenkisFile (Orquestador de la prueba)
2. Tener en cuenta:
   1. Diseño de scripts (Definición, Contextualización)
      1. Nombre sin caracteres, todo en camelCase o guion bajo
      2. Enumerar de forma consecutiva, adicionar el nombre del servicio con un guion que lo identifica.
      3. Los listener:
         1. Árbol de resultados
         2. Gráfico
         3. Escritor de datos simple
         4. Backend Listener
      4. Validar a nivel local que funcione, quitar los listeners
      5. Usar rulas relativas
      6. Colocar el BeanShell de la creación de documento: Log\_(Nombre).txt
      7. No dejar nombre default, si se va a extraer, dejar el nombre con esa transacción
3. Configuración YML:
   1. Por cada escenario que haya: BaseLIne, Carga y Estrés
      1. BaeLine:
         1. concrrency: 1
         2. hold-for: 9 (Mantener contante la concurrencia)
         3. ramp-up: 1m
         4. Mitad usuarios concurrentes, Mitas usuario constantes
      2. Carga:
         1. concrrency: 38
         2. hold-for: 25 (Mantener contante la concurrencia)
         3. ramp-up: 15m
         4. script que se va a llamar: (NombreTransacción).jmx
         5. intervalo: 5s (Cada cuándo guarda información)
      3. passfail.yml: Contiene los RNF
   2. JenkisFile-BaseLine, cada uno debe tener el .yml de los escenarios:
      1. Orquestador del proceso de ejecución, crear el repo en GitHub y colocar el link del repo dentro del Jenkins file.
4. Ejecución:
   1. Creación de Pipeline en una ruta especifica con la carpeta de Jenkins
   2. Creación en GitHub
   3. Ejecución de BaseLine por defecto, el estrés y la carga se hace a demanda
5. [Conectar con Grafana los resultados de las pruebas]