* Commandes de base de shell
* Utilisation d’une machine virtuelle : Oracle VM Virtual Box, MobaXterm
* Introduction JMeter

* Le plan de la présentation :
  + Les enjeux de performances et la problématique
  + Temps de réponse
  + Le cadre technique
  + Organisation d’un projet de performances
  + L’analyse des mesures
* Les enjeux de performances et la problématique :
  + Temps de réponse
  + Consommation de CPU
  + Le débit (par exp combien de linge traiter en même temps)
  + Nombre d’utilisateur
  + Stabilité de l’application : combien de temps l application fonctionne sans interruption
* Temps de réponse :
  + C’est la durée écoulée entre le début de l’émission de la requête et la fin de la réception de la réponse.
  + Le Frist Byte : le premier octet de la réception de la réponse.
  + Est en plusieurs étapes :
    1. Transformer la requête en adresse
    2. Etablissement de connexion avec le serveur : temps réseau est reparti
       - Temps interface :
         * chargement de temps de réponse,
         * mise en forme de la réponse,
         * Les critères impacte le temps de réponse : le navigateur, le device (os, hardware), le niveau de la charge d la batterie
       - Les équipements réseau: routeur, firewall
       - Les moyens de communication : câble, antenne
    3. Traitement applicatif de la requête sur le serveur :
       - Serveur web http : Apache, NGINX
       - Serveur application : JBoss (s’effectue les traitements lourd)
       - Serveur base de donnes : Oracle
    4. Envoi de la réponse
    5. Restitution de la réponse au niveau du device (n’est pas pris en compte dans le test de charge)
* Etapes d’une compagne de test de performances :
  1. **Spécification** (cadrage):
     + Prise en compte des exigences
     + Définition des objectifs
     + Choix des scénarios
     + Stratégies
  2. **Préparation**:
     + Provisionnement de l’infra
     + Installation du monitoring
     + Génération des données
     + Scripting : permet l’automatisation d’un scénario de test : Génération d’un script qui va reproduire les requêtes
       - Corrélation des requêtes
       - Points de mesures
       - Point de contrôle
     + Création des scénarios de test (3 scénarios max)
  3. **Exécution**:
     + Lancement des tirs (au fur a mesure)
     + Rédaction des rapports pour le client après chaque tire
     + Analyse des résultats
     + Conclusions par rapport aux objectifs
  4. **Restitution**:
     + Bilan de compagne
* Environnement de test :
  + Application
  + Outils de monitoring
  + Données de test
  + Infrastructure
* Platform de test :
  + Jeux de données
* Chaque problème est unique et sa solution est unique
* Les objectifs, le modèle de charge, et les mesures sont spécifiques à chaque compagne de test.
* Le SAT : le Système à tester : ses éléments et ses limites. Qu’et ce qu’on va tester ?
  + Il faut définir le plus petit SAT permettant d’atteindre les objectifs
  + Le SAT va impacter le modèle en charge et le choix des métriques
  + Les bénéfices de la définition d’un SAT :
    - Comparaison facile des résultats
    - Reproduction des tirs plus rapides
    - Meilleurs contrôle de l’environnement et mois d’interlocuteur
  + Lister les différents services et fonctionnalités du SAT
    - Par exemple une application web utilise des requêtes http et une Base de données utilise des requêtes SQL
  + Lister les différents résultats obtenus du SAT => Analyse des résultats
  + Les bénéfices de la définition des fonctionnalités et les lister les résultats :
    - Sélection des bonnes métriques
    - Construction du modèle de charge (fonctionnalités et jeux de données)
  + Comment choisir les choix services à tester dans le SAT :
    - Les traitements les plus utilisés
    - Les traitements critiques : paiement par exemple
* Les métriques : qu’est ce qu’on va mesurer ?
  + Elles représentent les critères d’évaluation des performances du SAT
  + Elles sont réelles :
    - La vitesse
    - Le taux d’erreur
    - La disponibilité de l’application, du serveur,
    - Stabilité
    - Temps de réponse
    - Le débit
  + Le bon choix des métriques permet une évaluation correcte d’atteinte des objectifs (Résultats vs exigences)
* Les paramètres : Qu’est ce qui impacte les performances ?
  + 2 catégories :
    - Paramètres système : hard et soft
    - Paramètres charge : lie à l’activité de l’utilisateur
  + Facteurs de performances : ensemble de paramètre qui vont varier pendant le test
  + Chaque facteur possède plusieurs niveaux
  + On commence avec un min de facteurs et de niveaux puis identifier ceux ayant le plus d’impacte
  + Les bénéfices :
    - Limiter les tests
    - Faciliter l’analyse des résultats
* Le modèle de charge : les scénarios
  + Il doit être représentatif de l’utilisation du système (représentation des services pertinents)
  + Il doit se baser sur le comportement du système en production ou ce qui est attendu
  + Bénéfices :
    - Tester des scénarios pertinent au regard de ce qui se passera en production
* Analyse et interprétation des résultats :
  + Les résultats obtenus sont de nature aléatoire donc les tirs
  + La variabilité des résultats doit être prise en compte
  + La seule comparaison des moyennes peut emmener à des erreurs d’interprétation
  + Les résultats fournissent la base de l’analyse mais pas des conclusions
* Présentation des résultats :
  + Par des graphes
  + Présentation claire de résultats
  + Préconisation
* LoadRunner : utilisation locale utilisé par dev
* HP performance center : utilisation par chaque équipe pour dérouler les tests du sprint en environnement dédié
* Jenkins : ordonnanceur
* **Les erreurs à éviter** :
  + Définition pas claire des objectifs
  + Faire une analyse sans une bonne compréhension du problème
  + Mauvais choix des métriques, ou l’oublie de métriques
  + Modèle de charge inapproprié
  + Omission de paramètres : CPU, RAM, charge …
  + Erreur d’analyse : insuffisance de données
  + Absence d’analyse
  + Négliger des valeurs d’écart : elles peuvent refléter un réel problème et doivent être analysé
  + Ne pas anticipé l’évolution du système (par exemple la taille des bases de données)
  + Analyse trop compliqué
  + Mauvaise prise en compte de la variabilité (ne pas utiliser uniquement la moyenne)
  + Mauvaise présentation des données (elles doivent aider à la prise de décision)
  + Oublier de décrire les conditions dans lesquelles a été réalisée la compagne
* **Le scripting**
* **ALM (Application Life Management) :**
  + Quality Center
  + Performance
* Il faut avoir un modèle d’usage de l’application en se basant principalement sur deux 2 critères :
  + Nombre de VU (virtual user)
  + Rythme applicatif (connaitre les limites de l’application)
* **Utilisation NeoLoad**
  + Variable sont crée par le testeur, elles représentent généralement le jeu de données
  + Les paramètres dynamiques sont générés automatiquement par l’application. La procédure de corrélation consiste à récupérer le paramètre dynamique
  + Une itération est rejoue complet dans le parcours d’utilisateur
  + Pacing par VU: 360 secondes / (nombre d’acte métier / heure / VU)
  + Il faut sauvgarder toujours la premiere capture
  + Test Unitaire en perfoamance == test pour un seul user
* **LoadRunner** :
  + VUG : Virtual User Generator : génération du cas de test dans un script
  + Controller : exécution du scenario de test
  + Analyser : analyse des résultats du Controller sous forme de graphes et de rapport
  + Le protocole Citrix ?
  + Le pacing dasn LoadRunner :
    - Démarrer une nouvelle itération directement après la fin de la précédente
    - Démarrer une nouvelle itération après (x secondes) de la fin de la précédente
    - Démarrer une nouvelle itération dans un intervalle fixe tout les x seconde