



Checklist pro Performance Testera

by Canarytrace

Checklist pro Performance Testera

Checklist slouží pro performance testera pro jasný seznam činností aktivit v rámci test designu, test exekuce, vyhodnocování a reportování PT. Stejně tak může dobře posloužit dalším zúčastněným osobám jako jsou vývojáři a architekti, kteří pomáhají jako garanti s dodávkou informací ze svých oblastí v rámci plánování, test designu, investigace a vyhodnocování.



Příprava Performance Testu

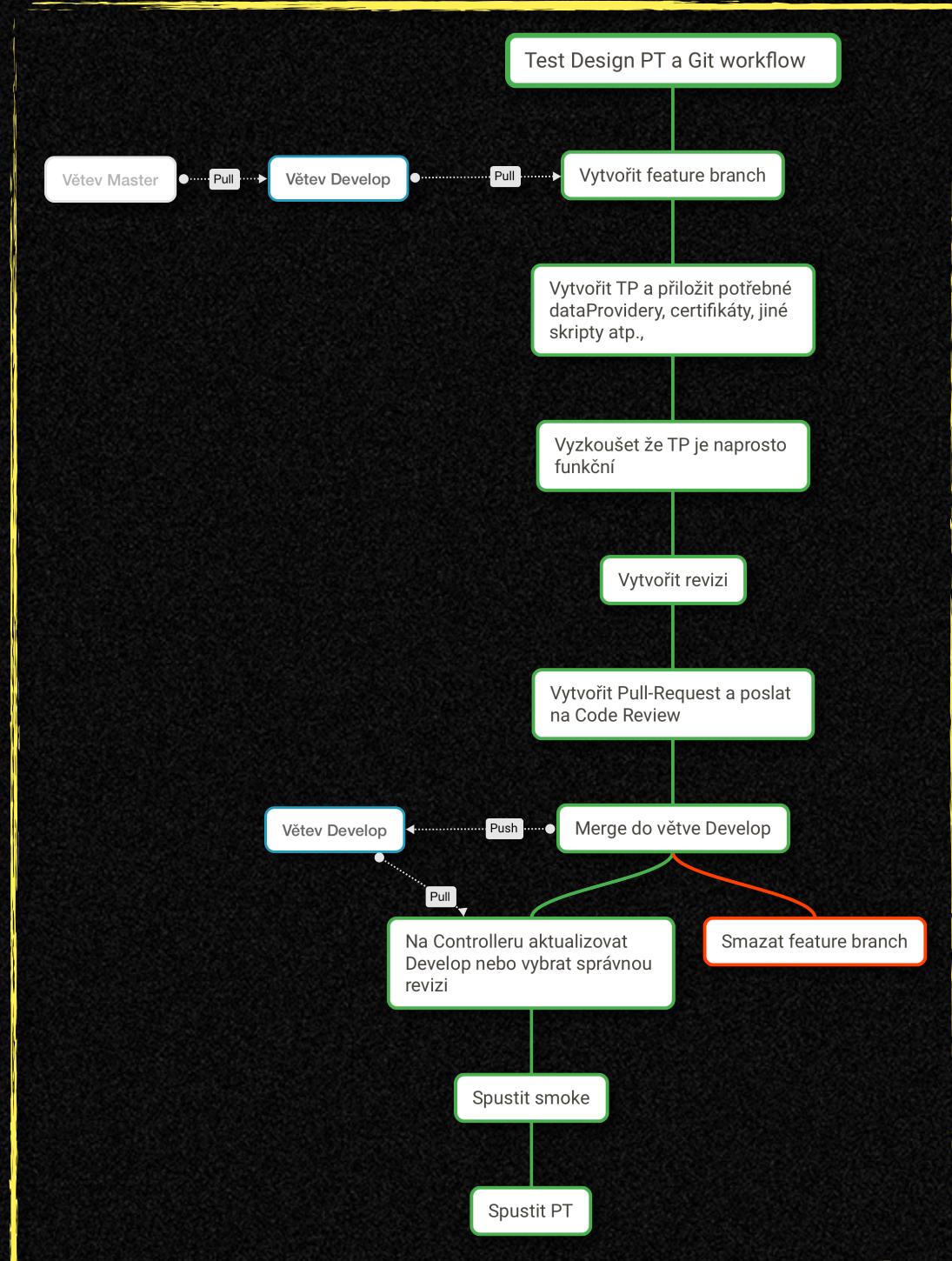
Krok	Oblast	Popis
1	Úvodní schůzka	Sběr požadavků a rizik pro sestavení TPA za účelem úspěšného provedení PT.
2	PTA	PTA je vypracována a schválena (např. zadavatelem a architektem). Vzhledem k úspoře času se nemusí čekat na kompletní dokončení PTA, ale může se začít s vývojem doprovodných nástrojů, přípravou testovacích dat a zároveň může vznikat i raná podoba PT, na kterém lze odzkoušet možná rizika.
3	Test design	Test plán je hotov a prošel Code Review. Číslo revize v gitu je zaznamenáno do PTA.
4	Smoke	Test plán je nasazen na generátor a generovaná zátěž je ve WM snížena na co nejmenší možné hodnoty a nechá se rotovat oproti cílovému testovacímu prostředí. Smoke test má i při zátěži 1VU a dlouhodobé rotaci velký význam a přínosy. Poukazuje na maximální možnou propustnost komponenty nebo celého testovacího prostředí a na jeho stabilitu. Může přinést informace o výpadcích či kapacitních problémech. Zároveň prověřuje samotný test plán, jestli správně funguje.
5	Monitoring	Monitoring dotčených systémů funguje, máme přístupy do logů a logování funguje. Umíme interpretovat grafy a vyčítat logy.

PT může vznikat již během vývoje

Malé PT zpravidla komponentové úrovně může vytvářet i vývojář a používat je při vývoji aplikace, nebo je může vytvářet dedikovaný tester, který je následně jako znova použitelnou komponentu či test fragment použije pro konstrukci **velkého PT** pro test exekuci **E2E PT**.



Test Design PT a Git Workflow



Krok	Oblast	Popis
1	Feature Branch	<p>Nikdy nevytváříme PT v hlavní větvi, ale vždy vytvoříme vlastní feature branch, která může mít následující syntaxi:</p> <ul style="list-style-type: none"> • příjmení-číslo ticketu nebo název projektu • příjmení-téma <p>Například:</p> <ul style="list-style-type: none"> • panek-cashloan • panek-1234 • panek-fix-oauth <p>Postup vytvoření feature branche:</p> <ol style="list-style-type: none"> 1. jsem na větvi <u>develop</u> => <u>git pull</u> 2. <u>git checkout -b panek-cashloan</u> 3. jsem přepnuty na nový větví <u>panek-cashloan</u> a můžu začít pracovat na Test Plánu <p>Feature branch vytvářím vždy, když vytvářím nový Test Plán nebo provádím úpravu existujícího Test Plánu, nebo při změně dataProvideru atp. Jednoduše řečeno, při jakékoli práci s Git repozitářem.</p>
2	Vytvořit Test Plán	<p>Vytvořit nebo upravit existující Test Plan v Gui SmartMeteru.</p> <ul style="list-style-type: none"> • V "Test Plan" komponentě do pole "Comments" uvedu link na TPA. • Pro vytvoření TP webové aplikace využijte: <ul style="list-style-type: none"> • Recorder pro nahrání TP (poté bude nahraný TP potřeba dočistit a doplnit chybějící skripty a assertace). • HAR z Google Chrome. • Generátor pro převod jiného procedurálního schématu na JMX schéma kompatibilní s Test Plány JMeteru, například export popisu integračních testů zdokumentovaných v Swagger. • WM namodelujte podle priorit business cases a podle produkčního profilu či odhadů od garantů. • Pro identifikaci průchodu uživatele použijte access logy nebo logy proxy serveru ve spolupráci s garantem. Mělo by být možné vysledovat průchod uživatele aplikací pomocí zaznamenané formy sessionID. • Nezapomeňte vložit před zvolené transakce odpovídající Think Time timer. • Do adresáře například "dataProviders" přiložte data provider, který je použit v Test Planu. Stejně doporučení platí pro keystore a další scripty. • Je potřeba ošetřit kompatibilitu cest pro platformy Windows a Linux. • Vyvarujte se přílišné komplexnosti či parametrizací Test Planu - čím jednoduší tím je přehlednější a údržba je snazší. • Pokud má Test Plan fungovat na více testovacích prostředích či při různé konfiguraci WM, kdy jiná konfigurace může znamenat jinou skladbu Controlleru, Transakcí či Sampleru, je potřeba vytvořit více Test Planu a všechny logické části vyexportovat a používat je v Test Plánech jako znova použitelné komponenty. Pokud některá z variant Test Planu přestane být potřeba, stačí jí smazat, ale znova použitelné logické komponenty zůstanou použité v dalších variacích Test Plánu. • Vytvářejte znova použitelné komponenty. • Transakce a Samplersy dobře pojmenujte. Transakce začínejte prvním velkým písmenem a Samplersy jsou psané vždy malým. Do názvu sampleru uvádějte i pořadí a o jaký typ requestu jde. • Před vytvořením revize nechte lokálně TP několikrát protočit s nastavením generované zátěže na minimum aby bylo jisté, že Test Plan funguje.



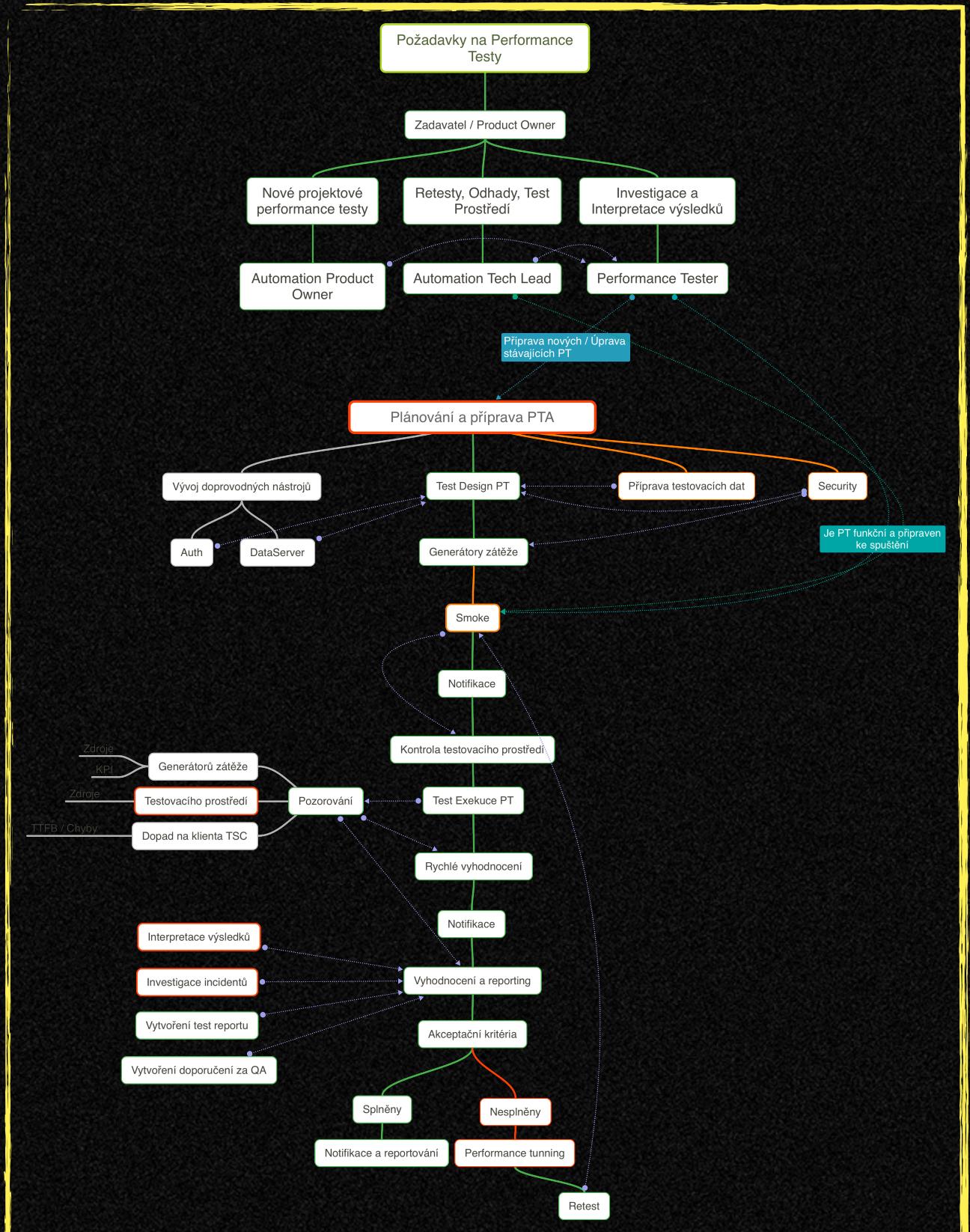
Krok	Oblast	Popis
3	Vytvořit revizi	<ul style="list-style-type: none"> Revizi vytvořte vždy pouze v okamžiku, kdy jste jste jisti, že Test Plán skutečně funguje. Do revize zahrňte dataProvidery, skripty, dokumenty, keystore, certifikáty atp. - vše co s novým Test Plánem nebo úpravou existujícího Test Plánu tématicky souvisí. Pokud se stane, že máte rozpracováno více tématicky nesouvisejících Test Plánu, skriptů, dataProviderů apod. Vytvořte pro každé z témat vlastní revizi. Pokud jde o konfiguraci, kterou používá více Test Plánu, použijte zvlášť revizi pro konfiguraci a zvlášť revizi pro Test Plán. Pokud by kolega pracoval na jiném Test Plánu a potřeboval by také stejnou konfiguraci, může pomocí cherry-picku zkopírovat potřebnou revizi do své feature branch. Do revize v žádném případě nepatří zakomentované kódy, vypnuté prvky Test Plánu pomocí (Disable či Toggle) či nepotřebné části Test Plánu, dataProvidery, skripty, konfigurace, certifikáty atp. Název revize může mít následující syntaxi: [Téma / projekt nebo oblast]: [krátký popis]: [added / changed / deleted / fix / typo] < prázdný řádek > Detailní popis obsahu a důvodu revize <p>Například:</p> <p><i>Cashloan:dataProvider:added</i></p> <p><i>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut enim ad minim veniam, quis nostrud exercitation ...</i></p> <ul style="list-style-type: none"> Název i popis revize pište v en. Vyvarujte se vytvářet názvy revizí typu (Upraveno, tohle už není potřeba, fix, oprava chyby atp.) Vyvarujte se použití jednopísmenných názvu revizí, jen ať máte revizi rychle vytvořenou. Pokud máte problémy s Gitem, požádejte o školení. Pomůžete tak sobě, kolegům a i vaším nástupcům. Vyvarujte se použití vizuálních správců Git repozitářů, práci s Gitem tak nemáte pod kontrolou a nevytváříte si silné návyky. Při nastavování CI s Performance Testy nebo při hledání pomoci na Stack Overflow budete stejně potřebovat Command Line rozhraní. Pokud pracujete ve své feature větvi, můžete používat --force --amend či interaktivní rebase pro úpravu revizí a historie revizí ve vaší feature větvi. V žádném případě nepřepisujte revize ve své feature větvi pokud jste vytvořili Pull-Request nebo pokud jsou vaše revize zmergovány do větve Develop. Pokud potřebujete vaši práci dodatečně upravit / opravit / nebo pro retest změnit konfiguraci WM TP, vždy vytvořte novou revizi. Snažte se do Gitu nepřidávat binární soubory, HTML reporty v .zip, nelze pak zobrazit diff. V žádném případě do Gitu nepatří velké soubory. Repozitáře tak významně naroste a bude se clonovat delší dobu. Práci na Test Plánu a vytvoření revize je možné ještě před dokončením kompletního Test Plánu podle PTA pokud se blíží termín dodávky PT.
4	Pull-Request	<ul style="list-style-type: none"> Pokud je z vašo pohledu Test Plán připravený, vytvořte Pull-Request a požádejte jiného performance testera, vývojáře či architekta nebo více osob dohromady o Code Review. Role kolegy, který provádí Code Review Test Plán je potřeba stáhnout a spustit. Zaměřte se na správné názvy Transakcí, Samplerů. Zaměřte se na assertace. • Zaměřte se na zadání a smysluplnost Test Plánu podle zadání, business požadavků a priorit v PTA. Zaměřte se na sémantiku Test Plánu, jak je postavený. Pokud je Code Review neúspěšný, konzultujte námítky a případné změny zahrňte do nové revize. Znovu požádejte o Code Review. Pokud je Code Review úspěšný, zamergujte svou feature větev do větve Develop a svou feature větev smažte včetně i ze vzdáleného serveru.



Krok	Oblast	Popis
5	Nasazení TP na Controlleru	<ul style="list-style-type: none"> Na serveru, kde se PT bude spouštět či na serveru odkud se budou řídit generátory záťěže aktualizujte vždy větev Develop. Pokud jde o spuštění smoke či PT zkонтrolujte, že je přepnuto na správnou revizi. Pokud jde o retest, správná revize je obsažena v PTA nebo v posledním Test Reportu.
6	Smoke	Před spuštěním Test Plánu ve smoke režimu je potřeba snížit nastavení WM na minimum a naplánovat rotaci Test Plánu (spouštět znova dokola) v cronu nebo v Jenkinsu.
7	Test exekuce PT	<ul style="list-style-type: none"> Při spuštění v distribuovaném módu je potřeba zkontolovat, že všechny zapojené generátory pracují. Dohledová test exekuce znamená sledovat stav nad probíhající test exekucí a sledování monitorovacích nástrojů. Zpravidla jde o důležité / kritické, komplexní TP nebo TP v TSC režimu. Bezdohledová test exekuce znamená spuštění testu na nastavenou dobu a nemusíme jej sledovat. Zpravidla se jedná o hodně jednoduché PT, nebo PT vytvářející Background Noise.



Performance Testing Life Cycle



Před spuštěním Performance Testu

Krok	Oblast	Popis
1	Notifikace	Zpravidla se support a další zúčastněné osoby informují s dostatečným časovým předstihem. Zároveň se komunikuje volné okno pro spuštění PT. Zároveň se může zamluvit místo pro tým, který řeší test exekuci, investigaci a vyhodnocení PT.
2	Kontrola Smoke Testu	<p>Smoke testy jsou funkční a nevykazují nefunkcionální či funkcionální defekty nebo incidenty.</p> <ul style="list-style-type: none"> • Prostupy fungují. • Test a jeho závislosti fungují. • Aplikace vrací očekávané odpovědi. • Před spuštěním PT vypnout smoky.
3	Test Plan (Performance / Stress test)	<p>Z gitu byla vybrána správná revize Test Plánu a všechny jeho závislosti jsou dostupné (certifikáty, data providery, znovu použitelné komponenty, data servery či auth. procesy.)</p> <p>Správnou revizi pro retest obsahuje předchozí Test Report.</p>
4	Kontrola testované aplikace, kontrola zdrojů a chování dalších dotčených systémů	<p>Testovaná aplikace je funkční a nevykazuje nefunkcionální či funkcionální defekty.</p> <ul style="list-style-type: none"> • Je nasazená správná verze. • Zdroje a konfigurace aplikačního serveru a infrastrukturních prvků odpovídají údajům v PTA. • Místo pro logy je dostatečné a logování funguje. • Grafy zdrojů dotčených systémů zobrazují klidový stav. <p>Pokud hodnoty zdrojů neodpovídají klidovému stavu, nelze spustit performance test. Zároveň je potřeba přihlédnout k běžné / očekávané zátěži = Backgroud Noise</p>
5	Kontrola stavů prostředí	<p>Před spuštěním PT může být vyžadováno, aby závislosti na testovaném ekosystému bylo splněno:</p> <ul style="list-style-type: none"> • Například denní life cycle joby systému. • Nebo nalítí potřebných dat do databáze atp.
6	Elasticsearch	<ul style="list-style-type: none"> • Prostupy fungují. • healthcheck ip:9200 Elasticsearch odpoví OK • cluster health curl -XGET 'ip:9200/_cluster/health?pretty' OK • node HEAP / CPU curl -XGET 'ip:9200/_cat/nodes?v' OK • Monitoring Elasticsearch ukazuje klidový stav.
7	Kontrola generátorů zátěže	<ul style="list-style-type: none"> • Procesy generátorů nejsou vytuhly. • Generátory mají prostupy do testovacího prostředí. • Generátory mají dostatek zdrojů. • Při využití IP Aliasingu je na generátorech dostupný seznam IP adres.
8	Přepnutí správné verze funkcionálních testů	<p>Pokud se jedná o PT webové aplikace, nebo PT generuje zátěž, která může ovlivňovat webovou aplikaci, je potřeba spustit WebPerformance Test meřící metriky v prohlížeči a dopad na klienta, když je aplikace pod zátěží. Stejně tak postupujeme s mobilní aplikací.</p>
9	Zkontrolovat běhové prostředí Selenia	<ul style="list-style-type: none"> • Stačí standalone mod, avšak Selenium Server be neměl vytěžovat jiný funkcionální test. • Screenshoty a DOM soubory lze ukládat. • Výsledky měření lze zapisovat do Elasticsearch.
10	Spustit WPT/Canary	<ul style="list-style-type: none"> • Ve smyčce. • PT spustit v době, kdy se funkcionální test alespoň 10x protočí. • Pokud funkcionální test naměří vysoké TTFB / Elapsed Time či aplikace je citelně pomalá, nemá smysl provádět test exekuci PT.



Během Performance Testu

Krok	Oblast	Popis
1	Test exekuce PT	<ul style="list-style-type: none"> V konfiguraci Localhost Controller (Gui) x (n) Remote Generator (nonGui) v případě dohledu. V konfiguraci Remote Controller (nonGui) x (n) Remote Generator (nonGui) v případě Performance Testu bez dohledu. Pro distribuovaný mód využít monitor jmx Kontrola SmartMeter logu, zdali test běží, zda je dostupný dataProvider nebo zda nevyexpirovala licence.
2	Monitoring SmartMeteru	V Kibaně ci na stdout spuštěné instance.
3	Monitoring nefunkcionálních metrik / KPI	V Kibaně
4	Monitoring aplikačních zdrojů a logů	
5	Monitoring chování webové aplikace	<ul style="list-style-type: none"> V Kibaně. Sledovat chování aplikace v prohlížeči, kde běží funkcionální test. Popis ze sledování lze dodat do výsledného Test Reportu.
6	Zastavení Performance Testu	<ul style="list-style-type: none"> Pokud jsou překročena stanovená SLA. Pokud testovaná aplikace vykazuje funkcionální či nefunkcionální incidenty nebo defekty nebo přestane odpovídat. Dosažení cíle = konce testu.
7	Zastavení funkcionálního testu	
8	Notifikace	Pro zadavatele a pro podporu prostředí s předběžnými výsledky PT.

Po skončení Performance Testu

Krok	Oblast	Popis
1	Stáhnout grafy a logy z monitorovacích nástrojů	<ul style="list-style-type: none"> Z agentů a monitorovacích aplikací, z prostředí pro správu databáze, proxyserveru, ze systémů pro nasazování a správu kontejnerových aplikací, které své hodnoty neukládají v Elasticsearch a tudíž se nezobrazují v Kibaně je potřeba vytvořit screenshoty ručně. (název serveru, časový rozsah, správně nastavené jednotky na osách X a Y) Stáhnout logy ze SmartMeteru pokud to dává smysl. Stáhnout log po paketové analýze během PT pokud byla provedena.
2	Investigace a vyhodnocení PT a vytvoření Test Reportu	<ul style="list-style-type: none"> Použít prázdnou šablonu Test Reportu a výsledný dokument uložit do namespace na Confluence k PT reportům či pod projekt. V Test Reportu se odkazujeme na PTA. Pro správnou investigaci je potřeba součinnost doménových specialistů a garantů. Do Test Reportu uvedeme "Manažerské shrnutí" což je HighLevel pohled nad proběhlým PT a nad splněním ACC. Do Test Reportu uvedeme "Doporučení za QA", kde doporučujeme jak a co znova otestovat, či změnit konfiguraci testovacího prostředí nebo poukážeme na rizika s neúspěšným PT.
3	Naplánování retestu	Pokud PT neprošel ACC, nebo průběh testu zastavil funkcionální či nefunkcionální incident či defekt, provede se investigace se založením ticketu. Ticket s incidentem se ukládá do reportu s popisem opravy nebo změny nastavení PT. Po BugFixingu nebo objasnění incidentu se naplánuje a provede retest.
4	Odeslání / Prezentace výsledků	



Tento checklist **upravte tak, aby plně vyhovoval** prostředí, kde se PT zabýváte. Doplňte kontaktní údaje zúčastněných osob PT Life Cycle (Release Manager, Architekti, Dev, Zadavatelé) emailové skupiny, slack channels, odkazy s přístupovými údaji na monitorovací nástroje, odkazy na reporty a schéma PT infrastruktury.



Radim Daniel Pánek
CEO, SDET & Performance Tester

Dlouhá léta pracoval jako vývojář a v posledních osmi letech se plně soustředí na výzkum v oblasti test automatizace. Působí v první linii od psaní testů přes správu infrastruktury pro automatizované testy až po evangelizaci test automatizace. Školí testery, navštěvuje různé QA týmy, mentoruje a hodnotí jejich test stack.

rdpanek@canarytrace.com
+420 731 011 200

<https://www.performance-testing.cz/>
<https://canarytrace.com>

